



**ПРОГРАММНЫЙ МОДУЛЬ ДЛЯ
ОБНАРУЖЕНИЯ ТРАНСПОРТНЫХ
СРЕДСТВ И ОПРЕДЕЛЕНИЯ
ХАРАКТЕРИСТИК ДОРОЖНОГО
ДВИЖЕНИЯ ПО ИЗОБРАЖЕНИЮ
ОТ ВИДЕОКАМЕРЫ**

**Программный модуль для
обнаружения транспортных
средств и определения
характеристик дорожного
движения по изображению от
видеокамеры
Руководство пользователя**

ЮФКВ.50030-01 95 01 К



НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

Module® является зарегистрированной торговой маркой ЗАО НТЦ "Модуль". Все другие торговые марки являются исключительной собственностью их соответствующих владельцев.

Содержание

1. Используемые сокращения	6
2. Общие сведения	7
2.1. Рекомендации по выбору, установке и настройке видеокамеры	9
2.1.1. Рекомендации по выбору видеокамеры	9
2.1.2. Размещение камеры	9
2.1.3. Ориентирование камеры	9
2.1.4. Поле наблюдения	11
2.1.5. Калибровка	12
2.2. Точность измерения	13
2.3. Защита от нелегального распространения	13
3. Использование TMKernel	14
3.1. Подключаемые модули и заголовочные файлы	14
3.2. Последовательность вызовов функций TMKernel	15
4. Описание TMKernel API	18
4.1. Типы данных, идентификаторы, перечисления и структуры	18
4.1.1. S_TMKER_Affixment	18
4.1.2. S_TMKER_Event	20
4.1.3. S_TMKER_Init	20
4.1.4. S_TMKER_LaneEvent	20
4.1.5. S_TMKER_Point	21
4.1.6. TMKER_DESC	22
4.1.7. TMKER_ERROR	22
4.1.8. TMKER_MAX_LANES	22
4.1.9. TMKER_MAX_VCLASSES	23
4.1.10. TMKER_VEHICLE_CLASS	23
4.2. Функции	23
4.2.1. TMKER_Create	23
4.2.2. TMKER_Destroy	24
4.2.3. TMKER_GetDongleLimit	24
4.2.4. TMKER_GetLibVersion	24
4.2.5. TMKER_Init	25
4.2.6. TMKER_ProcessFrame	25
4.2.7. TMKER_PrjToScreen	26
4.2.8. TMKER_Release	28
4.2.9. TMKER_ScreenToPrj	28
5. Описание AFFIXWIZARD API	30
5.1. Общие сведения	30
5.2. Структуры	37
5.2.1. S_AW_CamInternalParams	37
5.3. Функции	38
5.3.1. AW_LoadFromFile	38
5.3.2. AW_SaveToFile	38
5.3.3. AW_Show	39
6. Описание API для коррекции разметки	41
6.1. Общие сведения	41

6.1.1. Итерационная коррекция разметки	46
6.2. Структуры и идентификаторы	47
6.2.1. CA_HANDLE	47
6.2.2. S_CA_Camera	47
6.2.3. S_CA_Edges	47
6.3. Функции	48
6.3.1. CA_CorrectAffixment	48
6.3.2. CA_Create	49
6.3.3. CA_Destroy	49
6.3.4. CA_GetFramesForBkgr	50
6.3.5. CA_Init	50
6.3.6. CA_Operate	50
6.3.7. CA_Release	51
6.3.8. CA_Reset	52
6.3.9. CA_SetCameraParams	52
6.3.10. CA_SetImage	52
7. Работа в демонстрационном режиме	54
8. Приложение	55
8.1. Измерение характеристик дорожного движения с использованием TMKernel	55
8.1.1. Среднее значение и среднеквадратичное отклонение скорости	55
8.1.2. Занятость	56
8.1.3. Обнаружение останова ТС и движения по встречной полосе	56
8.1.4. Обнаружение дорожно-транспортной пробки	56

Список иллюстраций

2.1. Изображение дороги, соответствующее рекомендуемому положению камеры.....	10
2.2. Изображение дороги при предельном отклонении от вертикали	10
2.3. Степени свободы крепления камеры	11
2.4. Рабочая зона и Зона измерения занятости	12
4.1. Опорные точки	19
4.2. Границы рабочей зоны (для двух центральных полос движения)	19
4.3. Преобразование точки из проективной в экранную систему координат	27
4.4. Преобразование точки из экранных в проективные координаты	29
5.1. Задание параметров камеры	31
5.2. Задание опорных точек	32
5.3. Примеры выбора системы координат	33
5.4. Разметка рабочей области	34
5.5. Отображение результата разметки (реальный вид)	36
5.6. Отображение результата разметки (проекция)	37
6.1. Границы рабочей зоны до коррекции	42
6.2. Границы рабочей зоны после коррекции	42

1. Используемые сокращения

ТС - транспортное средство.

ЭК - электронный ключ.

ПД - петлевой детектор.

2. Общие сведения

Программный модуль TMKernel является библиотекой динамической загрузки в ОС Windows 2000/XP и служит для обработки последовательностей видеокадров с целью обнаружения ТС и определения характеристик дорожного движения. TMKernel предназначен для создания автоматических систем мониторинга и управления дорожным движением. Возможно использование модуля для обработки нескольких источников видеоданных на одной рабочей станции в рамках одного приложения.

Функции модуля позволяют получать характеристики дорожного движения по каждой из полос движения (до шести полос) в поле наблюдения. Для получения результатов обработки видеопоследовательности с заявленной точностью необходимо обрабатывать кадры последовательно и без пропусков. После обработки каждого кадра последовательности TMKernel формирует для каждой полосы структурированные данные со следующей информацией:

- Скорость обнаруженного ТС (км/ч)
- Длина обнаруженного ТС (м)
- Временной интервал между обнаруженным и предыдущим ТС
- Расположение обнаруженного ТС на изображении (X и Y координаты центра прямоугольника, описанного вокруг ТС)
- Класс обнаруженного ТС
- Признак нахождения ТС в [зоне измерения занятости](#)

Замечание

В данной версии значение длины следует использовать только как признак обнаружения ТС: если длина не равна нулю, значит обнаружено ТС. Расстояние между ТС - это расстояние между бамперами, причем для ТС, движущихся навстречу камере, это расстояние между передними бамперами, для ТС, движущихся от камеры, это расстояние между задними бамперами.

TMKernel определяет следующие классы ТС:

- Мотоциклы
- Легковые автомобили
- Грузовые автомобили длиной до 11 метров
- Грузовые автомобили длиной от 11 до 14 метров
- Грузовые автомобили длиной свыше 14 метров
- Автобусы

TMKernel является программным детектором транспорта. Полученные от детектора данные могут быть использованы для получения характеристик дорожного движения или для выработки сигналов управления транспортным потоком.

Для правильной работы программного детектора необходимо выполнить процедуру калибровки видеокамеры. В данном руководстве термин «калибровка» понимается в расширенном смысле и предполагает, в том числе, определение внешнего ориентирования камеры относительно дорожного полотна. В результате калибровки

определяется разметка рабочей области детектора. Если положение и поле зрения камеры не изменяются, калибровка выполняется однократно.

Следует отметить, что алгоритм детектора транспорта работает с последовательностью кадров, а не с одиночным изображением. Важно также, чтобы видеокдры поступали без пропусков, желательно с темпом 25 кадров в секунду - для PAL камер и 30 кадров в секунду - для NTSC камер. В противном случае возрастут ошибки измерения.

Размерность изображения может быть любой (в разумных пределах), однако оптимальным следует считать формат CIF (352 x 288 или 384 x 288 для PAL камер и 320 x 240 для NTSC камер). При этом обеспечиваются достаточное качество обнаружения и невысокие вычислительные затраты.

Помимо детектора транспорта пакет TMKernel содержит дополнительные модули и утилиты:

- AFFIXWIZARD.dll - библиотечный модуль для калибровки камеры в интерактивном режиме (создание разметки)
- CorrectAffixmentDyn.dll - библиотечный модуль для автоматической коррекции разметки
- MakeAffixment.exe - программа для калибровки камеры в интерактивном режиме (создание разметки)
- MULTITHREAD.exe - демонстрационная программа обработки последовательности кадров из AVI файла
- TMKERNEL_VIDEO.exe - демонстрационная программа обработки последовательности кадров от видеокамеры
- Примеры использования библиотечных модулей и документация

2.1. Рекомендации по выбору, установке и настройке

2.1.1. Рекомендации по выбору видеокамеры

- Рекомендуется аналоговая цветная или монохромная камера
- Разрешение – 480 твл или выше
- Минимальная освещенность – 0.02 Лк или ниже
- Сигнал/шум – 48 дБ или выше
- Автоматическая диафрагма
- Возможность ручной установки минимального времени экспозиции 1/100
- Фокусное расстояние 4-8 мм для матрицы 1/3". Возможно использование варифокального объектива с автодиафрагмой

2.1.2. Размещение камеры

Видеокамеру следует размещать на горизонтальной ферме над проезжей частью (центральное расположение), либо на мачте дорожного освещения, расположенной рядом с проезжей частью (боковое расположение).

При центральном расположении камеры TMKernel может обрабатывать до шести полос движения (с заявленной точностью). При боковом расположении – до четырех полос.

- Высота расположения камеры: 8 – 20 м
- Рекомендуемая высота расположения камеры: 12 м
- Удаленность мачты освещения от края проезжей части при боковом расположении устройства: не более 3м

2.1.3. Ориентирование камеры

Для правильной работы необходимо расположить камеру так, чтобы:

- В поле зрения видеокамеры наблюдался участок дороги длиной не менее 50м;
- Изображение дороги на экране было максимально близко к вертикали;
- Линия горизонта не должна попадать в поле зрения видеокамеры.

На рисунке [2.1](#) показано изображение дороги, соответствующее рекомендуемому расположению камеры. На рисунке [2.2](#) показан предельный случай отклонения изображения дороги от вертикали. Изображение дороги не должно отклоняться от вертикали более чем на 30°.



Рисунок 2.1 - Изображение дороги, соответствующее рекомендуемому положению камеры

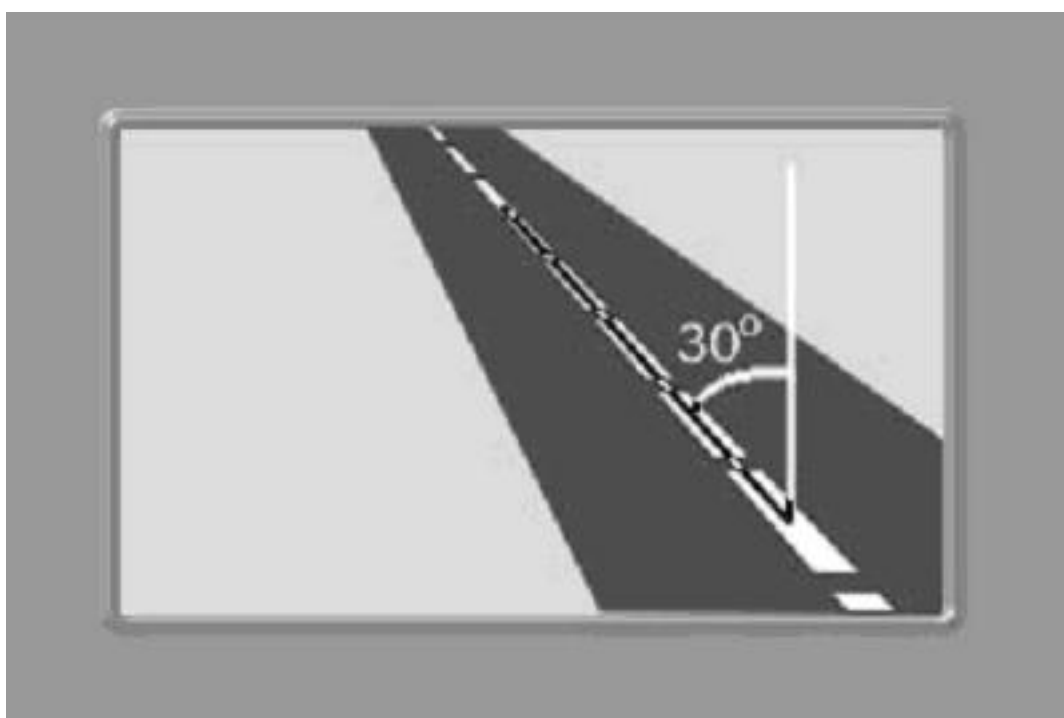


Рисунок 2.2 - Изображение дороги при предельном отклонении от вертикали

Крепление камеры должно иметь две степени свободы, позволяющие регулировать ориентацию камеры в направлениях, показанных на рисунке [2.3](#).

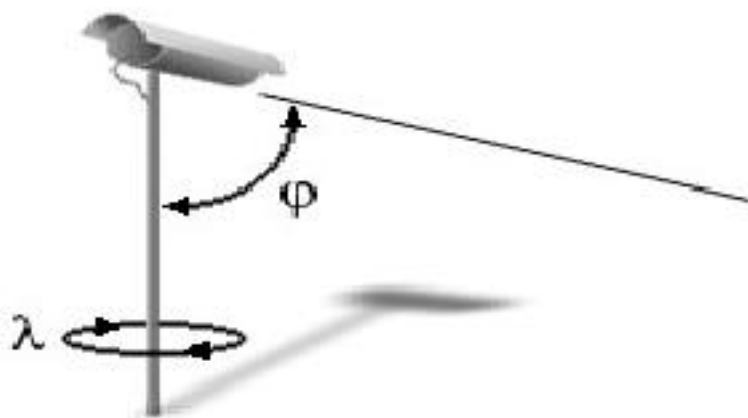


Рисунок 2.3 - Степени свободы крепления камеры

При настройке допускается изменение угла наклона камеры и вращение вокруг вертикальной оси.

2.1.4. Поле наблюдения

Видеокамера устанавливается над наблюдаемым участком дороги. На этом участке в результате калибровки задаются (размечаются) следующие зоны:

2.1.4.1. Рабочая зона

Ширина – от 3.2 до 19.2 м, длина – от 20 до 30 м. TMKernel обрабатывает только область изображения, находящуюся внутри *Рабочей зоны*. В зависимости от размещения, одна камера может отслеживать участок дороги шириной до 6 полос движения (при центральном расположении над дорогой) или до 4 полос (при боковом).

2.1.4.2. Зона сопровождения

Соответствует одной полосе движения на дороге. Возможно задание до 6 *Зон сопровождения* в одной *Рабочей зоне*.

2.1.4.3. Зона измерения занятости

Зона измерения занятости используется для вычисления параметра транспортного потока “Занятость” и расположена в конце каждой *Зоны сопровождения*. Длина *Зоны измерения занятости* составляет 5 метров, ширина равна ширине полосы *Зоны сопровождения*.

На рисунке [2.4](#) показана *Рабочая зона* (обведена зелёным цветом) и *Зоны измерения занятости* (красные прямоугольники).

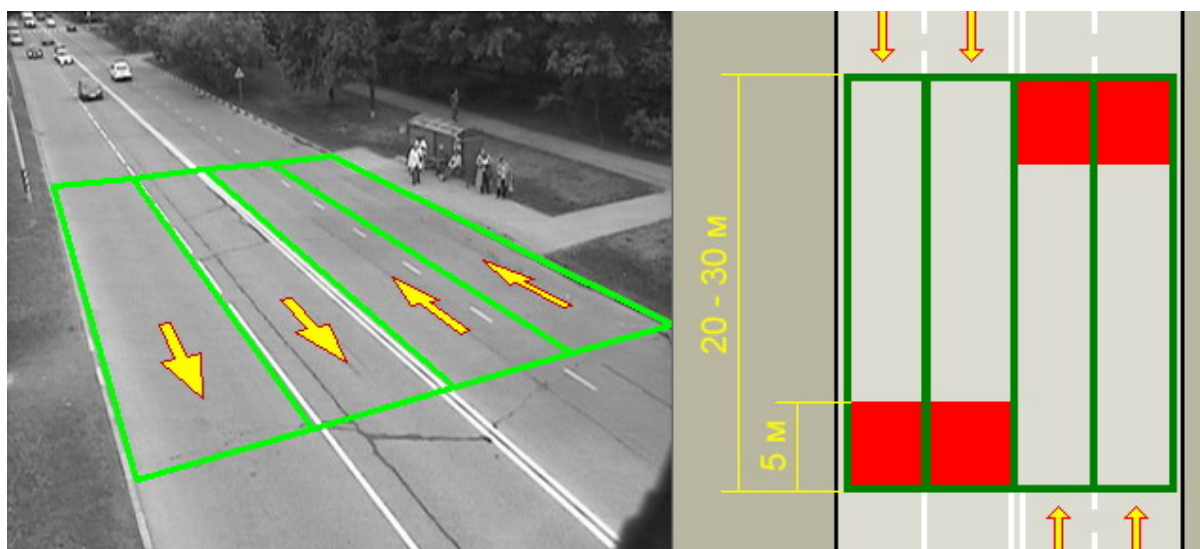


Рисунок 2.4 - Рабочая зона и Зона измерения занятости

2.1.5. Калибровка

Калибровка - это привязка камеры к наблюдаемой сцене и разметка рабочей зоны детектора. Возможны два варианта калибровки.

Если характеристики камеры известны (фокусное расстояние, размер матрицы, число эффективных пикселей), то процедура достаточно проста: на изображении нужно отметить границы анализируемого участка дороги. TMKernel сам определит необходимые параметры разметки рабочей зоны детектора.

Если характеристики камеры не известны, то потребуется выполнить некоторые действия на дороге. Прежде всего, надо выбрать в поле зрения видеокамеры четыре опорные точки. Опорная точка – малоразмерный объект, хорошо поддающийся идентификации на изображении. Опорными точками могут служить, например, метки, нанесенные краской на проезжей части. Необходимо измерить координаты всех опорных точек в плоской декартовой системе координат, связанной с полотном дороги и с произвольно выбранным началом отсчета. Ось Y системы координат направлена вдоль дороги, ось X - поперек. На одной прямой не должны находиться более двух опорных точек. Опорные точки должны находиться на максимальном (по возможности) расстоянии друг от друга, но при этом всё же находится в поле зрения видеокамеры. Чем больше удалены опорные точки, тем точнее будет выполнена привязка к сцене.

Наличие опорных точек в поле зрения камеры требуется только на этапе разметки.

Далее в документации описано, как при помощи [“мастера разметки”](#) осуществляется задание границ анализируемого участка дороги и ввод информации об опорных точках .

2.2. Точность измерения

Относительная погрешность определения характеристик дорожно-транспортной обстановки при выполнении рекомендаций по параметрам, установке и настройке камеры не превышает:

- 5% – Число ТС
- 10% – Средняя скорость
- 10% – Средняя дистанция
- 10% – Классификация ТС

Условия измерения:

- видимость ТС или фар ТС (ночью) - не менее 50 м
- скорость ТС - более 20 км/ч
- статистическая выборка - не менее 1000 ТС .

2.3. Защита от нелегального распространения

TMKernel защищён от нелегального распространения электронным ключом (ЭК). Для работы с полнофункциональной версией модуля необходимо присоединить прилагаемый ЭК к USB порту компьютера. При отсутствии ЭК TMKernel будет работать в демонстрационном режиме (см. раздел [работа с демонстрационной версией](#)).

Для работы с мастером разметки ЭК не требуется.

3. Использование TMKernel

3.1. Подключаемые модули и заголовочные файлы

Для работы с программным модулем необходимо включить файл `TMKERNEL.dll` в область "видимости" операционной системы. Рекомендуется создать переменную среды окружения, например, с именем `TMKernel`, в которой указать путь к каталогу дистрибутива и добавление в переменную окружения `PATH` строки `%TMKERNEL%\bin`.

Каталог дистрибутива содержит 5 подкаталогов:

- *bin* - содержит динамически подключаемые библиотеки `TMKERNEL.dll`, `AFFIXWIZARD.dll`, `CorrectAffixmentDyn.dll`, а также пример `MULTITHREAD.exe`, демонстрационную программу обработки видео `TMKERNEL_VIDEO.exe` и программу для создания и модификации дорожной разметки `MakeAffixment.exe`.
- *doc* - содержит файл справки `TMKERNEL.chm`
- *include* - содержит заголовочные файлы `TMKERNEL.h`, `AFFIXWIZARD.h` и `CorrectAffixment.h` с описанием функций библиотек
- *lib* - содержит файлы `TMKERNEL.lib`, `AFFIXWIZARD.lib` и `CorrectAffixmentDyn.lib` для раннего связывания
- *examples* - примеры использования библиотек

Для использования библиотеки в C/C++ программах включите в свой исходный файл директиву `#include "TMKERNEL.h"`. Если используется среда разработки `MSVC++`, то для раннего связывания подключите к проекту файл `TMKERNEL.lib`, а если требуется вызывать мастер разметки, `AFFIXWIZARD.lib`. Используйте переменную окружения `TMKERNEL` для задания пути к файлам `TMKERNEL.h`, `AFFIXWIZARD.h`, `TMKERNEL.lib` и `AFFIXWIZARD.h`.

3.2. Последовательность вызовов функций TMKernel

В приведённом ниже листинге показан пример использования функций TMKernel в программе пользователя.

```

1.  extern BOOL GetFrame(BYTE *pbFrame);
2.  extern BOOL GetAffixment(S_TMKER_Affixment *pAffix);

3.  static const int FRAME_WIDTH = 352;
4.  static const int FRAME_HEIGHT = 288;
5.  static const int FRAME_RATE = 25;

6.  void main(){
7.      BYTE *pbFrame = NULL;
8.      S_TMKER_Init Init = {0};
9.      TMKER_DESC TMK = NULL;
10.     S_TMKER_Event Event = {0};
11.     int i;

12.     try{
13.         if(!GetAffixment(&Init.Affix)){
14.             throw "No road affixment";
15.         }
16.         Init.Affix.nFrameWidth = FRAME_WIDTH;
17.         Init.Affix.nFrameHeight = FRAME_HEIGHT;
18.         Init.Affix.nFrameRate = FRAME_RATE;

19.         if(!(pbFrame = new BYTE[FRAME_WIDTH * FRAME_HEIGHT])){
20.             throw "Memory allocation";
21.         }

22.         if(TMKER_ER_OK != TMKER_Create(&TMK)){
23.             throw "TMKER_Create()";
24.         }

25.         if(TMKER_ER_OK != TMKER_Init(TMK, &Init)){
26.             throw "TMKER_Init()";
27.         }

28.         while(GetFrame(pbFrame)){
29.             if(TMKER_ER_OK != TMKER_ProcessFrame(TMK, pbFrame, &Event)){
30.                 throw "TMKER_ProcessFrame";
31.             }
32.             for(i=0; i<Init.Affix.nLanes; i++){
33.                 if(Event.Events[i].nVehicleLength){
34.                     printf("Event on lane %d\n", i + 1);
35.                     switch(Event.Events[i].nVehicleClass){

```

```

36.         case TMKER_VC_MOTORCYCLE:
37.             printf("\tClass: motorcycle\n");
38.             break;
39.         case TMKER_VC_CAR:
40.             printf("\tClass: car\n");
41.             break;
42.         case TMKER_VC_STRUCK:
43.             case TMKER_VC_MTRUCK:
44.                 printf("\tClass: truck\n");
45.                 break;
46.         case TMKER_VC_BUS:
47.             printf("\tClass: bus\n");
48.             break;
49.         case TMKER_VC_LTRUCK:
50.             printf("\tClass: trailer\n");
51.             break;
52.     }
53.     printf("\tSpeed: %d\n", Event.Events[i].nVehicleSpeed);
54.     printf("\tLenght: %d\n", Event.Events[i].nVehicleLength);
55.     printf("\tHeadway: %d\n", Event.Events[i].nVehicleHeadway);
56.     printf("\tX pos: %d\n", Event.Events[i].nVehicleOnScreenX);
57.     printf("\tY pos: %d\n", Event.Events[i].nVehicleOnScreenY);
58. }
59. }
60. }
61. catch(char *szError){
62.     printf("Error: %s\n", szError);
63. }
64. TMKER_Release(TMK);
65. TMKER_Destroy(TMK);
66. delete [] pbFrame;
67. }

```

В строке 1 экспортируется функция для получения одного видеокadra. Функция принимает на вход буфер для кадра. Предполагается, что каждый вызов GetFrame заполняет буфер pbFrame новым кадром из последовательности видеокadров. Возвращает булево значение: TRUE - кадр успешно получен, FALSE - кадр отсутствует. Размер кадра задаётся в строках 3 и 4. Для работы с TMKernel требуется кадр с одним байтом на пиксель (256 градаций серого). В буфере pbFrame кадр размещается в правильном порядке, то есть первая строка кадра располагается в начале буфера. Функция GetFrame должна быть реализована пользователем и в листинге не приводится.

В строке 2 экспортируется функция заполнения структуры для разметки зон детектора. Функция принимает на вход указатель на структуру разметки (см. [S_TMKER_Affixment](#)), которая должна быть заполнена. Возвращает булево значение: TRUE - данные для разметки успешно получены, FALSE - данные отсутствуют. Данные для заполнения структуры могут быть считаны из файла, или получены с помощью мастера разметки.

Функция `GetAffixment` должна быть реализована пользователем и в листинге не приводится.

В строках 3, 4, 5 задаются размеры видеокадра и количество кадров в секунду.

7 - 11: объявляются и инициализируются необходимые переменные.

12: начало блока `try - catch` для обработки ошибок.

13 - 18: заполнение структуры разметки детектора для инициализации `TMKernel` (см. [S_TMKER_Init](#)).

19 - 21: выделяется память для буфера видео кадра.

22 - 24: вызов [TMKER_Create](#) - для создания экземпляра `TMKernel` и получения его описателя.

25 - 27: вызов [TMKER_Init](#) - для инициализации экземпляра детектора.

28: организация покадрового цикла. Условием выхода из цикла является отсутствие кадра.

29 - 31: вызов [TMKER_ProcessFrame](#) - для обработки одного кадра.

32: начало цикла по всем размеченным полосам движения.

33 - 56: вывод информации об обнаруженном ТС. Наличие ТС на данной полосе движения определяется ненулевым значением поля `nVehicleLength` структуры [S_TMKER_LaneEvent](#).

62: вывод сообщения об ошибке.

64 - 66: уничтожение экземпляра детектора и освобождение занимаемых ресурсов.

4. Описание TMKernel API

4.1. Типы данных, идентификаторы, перечисления

4.1.1. S_TMKER_Affixment

Разметка зон детектора

```
typedef struct tagS_TMKER_Affixment{
    int nLanes;
    int nDirection[TMKER\_MAX\_LANES];
    int nLanesWidth[TMKER\_MAX\_LANES];
    S\_TMKER\_Point pptImage[4];
    S\_TMKER\_Point pptEarth[4];
    S\_TMKER\_Point pptLeftRoadEdge[2];
    S\_TMKER\_Point pptRightRoadEdge[2];
} S_TMKER_Affixment;
```

nLanes - количество анализируемых полос движения.

nDirection - направление правильного движения для каждой полосы. 1 - ТС движутся к камере, 0 - ТС движутся от камеры.

nLanesWidth - ширина полосы в процентах относительно ширины рабочей зоны. Сумма значений *nLanesWidth* для всех анализируемых полос должна быть равна 100. При передаче структуры в функцию [TMKER_Init](#) можно установить нулевые значения для всех полос. В этом случае ширина всех анализируемых полос будет одинаковой.

pptImage - координаты опорных точек на изображении (экранные координаты) (пикс).

pptEarth - координаты опорных точек в земной системе координат (см).

pptLeftRoadEdge - координаты начала/конца отрезка, лежащего на левой границе рабочей зоны (на изображении, экранные координаты) (пикс).

pptRightRoadEdge - координаты начала/конца отрезка, лежащего на правой границе рабочей зоны (на изображении, экранные координаты) (пикс).

Начало системы координат на изображении находится в левом верхнем углу.

Допустимый диапазон *nLanes* - от 1 до [TMKER_MAX_LANES](#).

На рисунках показаны некоторые параметры разметки зон детектора.

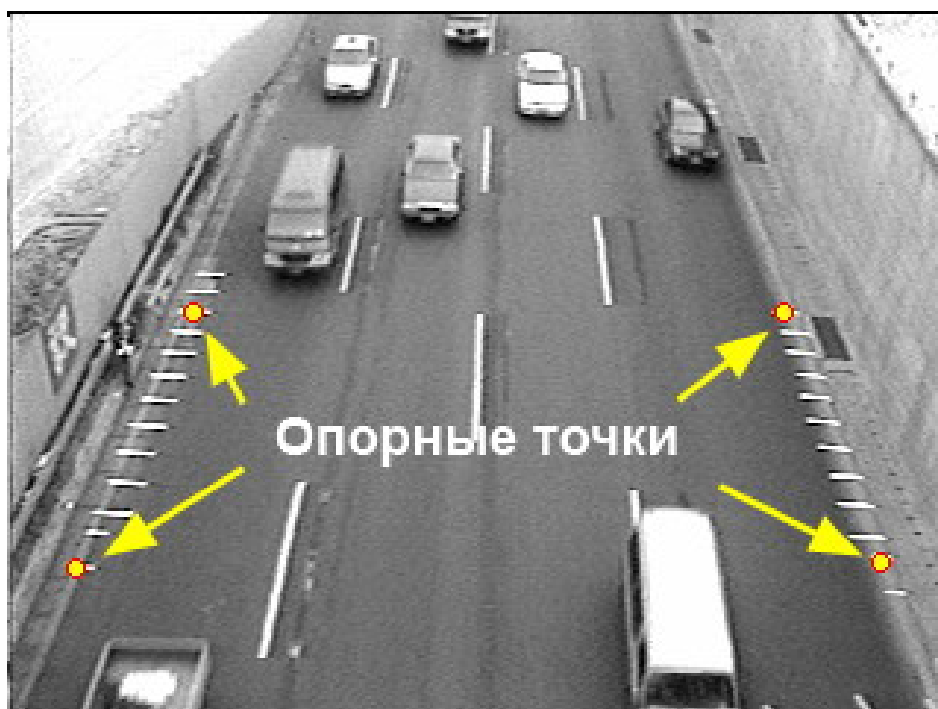


Рисунок 4.1 - Опорные точки

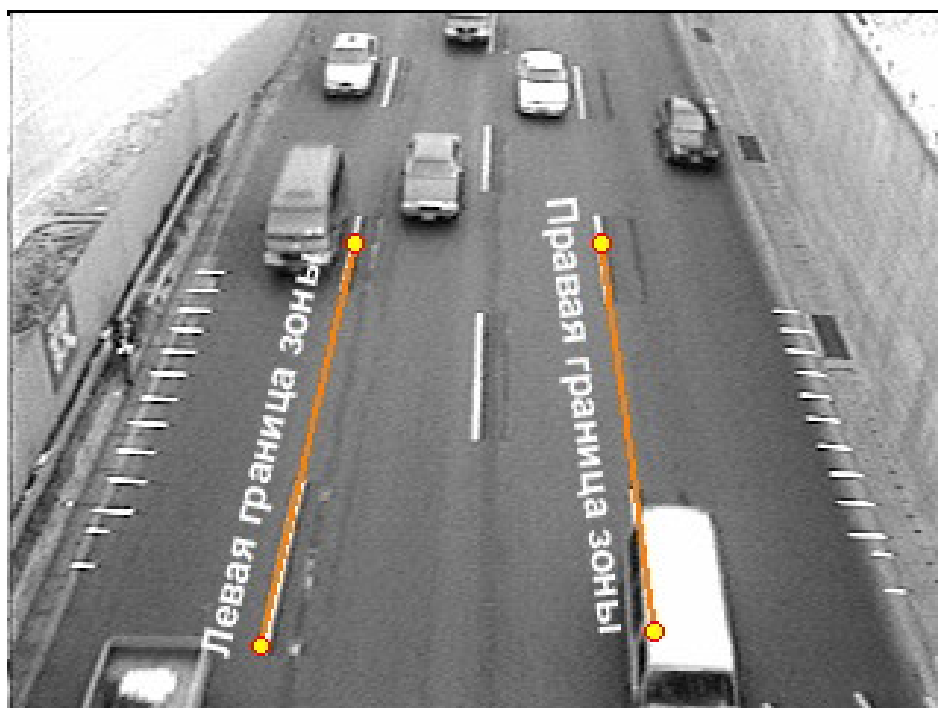


Рисунок 4.2 - Границы рабочей зоны (для двух центральных полос движения)

Опорные точки необходимо задавать в экранных и земных координатах. Порядок задания точек не важен, главное чтобы соблюдалось соответствие между экранными и земными

точками. Например, первая точка в массиве pptImage и первая точка в массиве pptEarth должны соответствовать одной опорной точке.

Объявлено в TMKERNEL.h

4.1.2. S_TMKER_Event

Покадровая информация об обнаруженных ТС для всех анализируемых полос в рабочей зоне

```
typedef struct tagS_TMKER_Event{
    S\_TMKER\_LaneEvent Events[TMKER_MAX_LANES];
} S_TMKER_Event;
```

Events - Покадровая информация об обнаруженных ТС для каждой из анализируемых полос в рабочей зоне.

Объявлено в TMKERNEL.h

4.1.3. S_TMKER_Init

Структура для инициализации детектора

```
typedef struct tagS_TMKER_Init{
    int nFrameWidth;
    int nFrameHeight;
    int nFrameRate;
    S\_TMKER\_Affixment Affix;
} S_TMKER_Init;
```

nFrameWidth - ширина обрабатываемых кадров (пикс)

nFrameHeight - высота обрабатываемых кадров (пикс)

nFrameRate - количество кадров в секунду обрабатываемого видео потока

Affix - структура для разметки зон детектора

Объявлено в TMKERNEL.h

4.1.4. S_TMKER_LaneEvent

Покадровая информация об обнаруженных ТС для одной дорожной полосы

```
typedef struct tagS_TMKER_LaneEvent{
    int nVehicleSpeed;
    int nVehicleLength;
    int nVehicleHeadway;
    int nVehicleOnScreenX;
    int nVehicleOnScreenY;
```

```

    TMKER_VEHICLE_CLASS VehicleClass;
    int nIsVehicleInDetectionArea;
} S_TMKER_LaneEvent;

```

nVehicleSpeed - скорость обнаруженного ТС (км/ч).

nVehicleLength - длина обнаруженного ТС (м).

nVehicleHeadway - временной интервал между обнаруженным и предыдущим ТС (с).

nVehicleOnScreenX - X-координата на изображении центра прямоугольника, описанного вокруг обнаруженного ТС (пикс).

nVehicleOnScreenY - Y-координата на изображении центра прямоугольника, описанного вокруг обнаруженного ТС (пикс).

VehicleClass - тип обнаруженного ТС.

nIsVehicleInDetectionArea - признак нахождения ТС в [зоне измерения занятости](#) на обработанном кадре. 0 - ТС отсутствует в зоне, 1 - ТС находится в зоне

Замечание

В данной версии значение длины следует использовать только как признак обнаружения ТС: если длина не равна нулю, значит обнаружено ТС. Расстояние между ТС - это расстояние между бамперами, причем для ТС, движущихся навстречу камере, это расстояние между передними бамперами, для ТС, движущихся от камеры, это расстояние между задними бамперами.

Начало системы координат на изображении находится в левом верхнем углу кадра.

Отрицательное значение *nVehicleSpeed* означает, что ТС движется в обратную от принятого направления сторону.

Значение *nVehicleSpeed* не может быть больше 255 км/ч или меньшим -255. Оно устанавливается равным 255 (-255) даже если реальная скорость ТС выше 255 км/ч (ниже -255 км/ч).

Равенство *nVehicleLength* нулю означает, что в данный момент детектор не обнаружил никаких ТС. В этом случае значения полей *nVehicleSpeed*, *nVehicleHeadway*, *nVehicleOnScreenX*, *nVehicleOnScreenY* и *VehicleClass* должны игнорироваться.

Объявлено в TMKERNEL.h

4.1.5. S_TMKER_Point

Координаты точки

```

typedef struct tagS_TMKER_Point{
    short shX;
    short shY;
} S_TMKER_Point;

```

shX - X координата

shY - Y координата

Объявлено в TMKERNEL.h

4.1.6. TMKER_DESC

Описатель экземпляра детектора

```
typedef int TMKER_DESC;
```

В одном процессе можно создать несколько экземпляров детектора транспорта, что позволяет обрабатывать видеопоследовательности от нескольких видеокамер. Все экземпляры работают независимо друг от друга. Для идентификации экземпляра необходимо передавать описатель детектора во все функции TMKernel, которые требуют этот описатель.

Объявлено в TMKERNEL.h

4.1.7. TMKER_ERROR

Идентификаторы ошибок

```
typedef enum tagTMKER_ERROR
{
    TMKER_ER_OK,
    TMKER_ER_INVALID_FUNC_PARAMETER,
    TMKER_ER_MEMORY_ALLOC,
    TMKER_ER_AFFIXMENT_INVALID,
    TMKER_ER_DONGLE_ERROR,
    TMKER_ER_NO_INIT
} TMKER_ERROR;
```

TMKER_ER_OK - нет ошибок

TMKER_ER_INVALID_FUNC_PARAMETER - неверный параметр функции

TMKER_ER_MEMORY_ALLOC - ошибка выделения памяти

TMKER_ER_AFFIXMENT_INVALID - ошибка при разметке детектора

TMKER_ER_DONGLE - ошибка при работе с электронным ключом

TMKER_ER_NO_INIT - не была вызвана функция инициализации [TMKER_Init](#)

Большинство функций TMKernel возвращают идентификатор ошибки.

Объявлено в TMKERNEL.h

4.1.8. TMKER_MAX_LANES

Максимальное количество полос в рабочей зоне

```
#define TMKER_MAX_LANES 6
```

Объявлено в TMKERNEL.h

4.1.9. TMKER_MAX_VCLASSES

Количество классов транспортных средств

```
#define TMKER_MAX_VCLASSES TMKER\_VC\_LTRUCK + 1
```

Объявлено в TMKERNEL.h

4.1.10. TMKER_VEHICLE_CLASS

Идентификаторы классов транспортных средств

```
typedef enum tagTMKER_VEHICLE_CLASS
{
    TMKER_VC_MOTORCYCLE,
    TMKER_VC_CAR,
    TMKER_VC_STRUCK,
    TMKER_VC_BUS,
    TMKER_VC_MTRUCK
    TMKER_VC_LTRUCK
} TMKER_VEHICLE_CLASS;
```

TMKER_VC_MOTORCYCLE - мотоцикл

TMKER_VC_CAR - легковой автомобиль

TMKER_VC_STRUCK - грузовой автомобиль, длиной до 11 м.

TMKER_VC_BUS - автобус.

TMKER_VC_MTRUCK - грузовой автомобиль, длиной от 11 до 14 м.

TMKER_VC_LTRUCK - грузовой автомобиль, длиной свыше 14 м. (трейлеры и автопоезда).

Объявлено в TMKERNEL.h

4.2. Функции

4.2.1. TMKER_Create

Создание экземпляра детектора транспорта

```
TMKER_ERROR TMKER_Create(TMKER\_DESC *pTMKernel);
```

pTMKernel - [out] указатель на описатель экземпляра детектора

Возвращаемое значение: идентификатор ошибки.

Описатель необходим для идентификации экземпляра детектора и используется как параметр для остальных функций TMKernel.

Для уничтожения созданного экземпляра необходимо вызвать [TMKER_Destroy](#).

Объявлено в TMKERNEL.h

4.2.2. TMKER_Destroy

Удаление экземпляра детектора транспорта

```
void TMKER_Destroy(
    TMKER\_DESC TMKernel
);
```

TMKernel - [in] описатель удаляемого экземпляра детектора

Уничтожает экземпляр детектора, который ранее был создан вызовом [TMKER_Create](#).

После вызова TMKER_Destroy не допускается дальнейшее использование описателя детектора. Рекомендуется после удаления детектора присвоить описателю нулевое значение.

Объявлено в TMKERNEL.h

4.2.3. TMKER_GetDongleLimit

Получение времени, оставшегося до истечения срока работы ЭК

```
TMKER\_ERROR TMKER_GetDongleLimit(
    unsigned long *pulDongleLimit
);
```

pulDongleLimit - [out] время, оставшееся до истечения срока работы ЭК (сек.).

Возвращаемое значение: идентификатор ошибки.

Функция может быть использована при эксплуатации ЭК с ограниченным сроком работы.

Объявлено в TMKERNEL.h

4.2.4. TMKER_GetLibVersion

Получение номера версии TMKernel

```
TMKER\_ERROR TMKER_GetLibVersion(
```

```
int *pMajor,
int *pMinor
);
```

pMajor - [out] старшее слово номера версии

pMinor - [out] младшее слово номера версии

Возвращаемое значение: идентификатор ошибки.

Старшее слово номера версии TMKernel изменяется при внесении существенных изменений, например, при изменении интерфейса.

Младшее слово версии TMKernel изменяется при внесении незначительных изменений, без изменения интерфейса.

Объявлено в TMKERNEL.h

4.2.5. TMKER_Init

Инициализация экземпляра детектора транспорта

```
TMKER_ERROR TMKER_Init(
    TMKER_DESC TMKernel,
    S_TMKER_Init *pInit
);
```

TMKernel - [in] описатель экземпляра

pInit - [in] указатель на экземпляр структуры для инициализации

Возвращаемое значение: идентификатор ошибки.

Функция инициализации должна быть вызвана до использования функций [TMKER_ProcessFrame](#), [TMKER_ScreenToPrj](#) и [TMKER_PrjToScreen](#).

Перед повторными вызовами TMKER_Init необходимо вызывать функцию [TMKER_Release](#).

Важно

Функция имеет особенности при работе в демонстрационном режиме (при отсутствии ЭК). Подробности см. в разделе [работа с демонстрационной версией](#).

Объявлено в TMKERNEL.h

4.2.6. TMKER_ProcessFrame

Обработка очередного видеокadra

```

TMKER\_ERROR TMKER_ProcessFrame (
    TMKER\_DESC TMKernel,
    unsigned char *pData,
    S\_TMKER\_Event *pEvent
);

```

TMKernel - [in] описатель экземпляра детектора

pData - [in] буфер с видео кадром

pEvent - [out] указатель на экземпляр структуры с информацией об обнаружении для всех полос движения

Возвращаемое значение: идентификатор ошибки.

Ширина и высота передаваемого кадра должны быть равны полям *nFrameWidth* и *nFrameHeight* структуры [S_TMKER_Init](#), которая была передана ранее в функцию [TMKER_Init](#). Для работы с *TMKernel* требуется кадр с одним байтом на пиксель (256 градаций серого). В буфере *pbFrame* кадр должен размещаться в правильном порядке, то есть первая строка кадра располагается в начале буфера.

Функцию *TMKER_ProcessFrame* необходимо вызывать при обработке каждого кадра. Пропуск кадров не допускается.

Объявлено в *TMKERNEL.h*

4.2.7. TMKER_PrjToScreen

Преобразование точки из проективной в экранную систему координат

```

TMKER\_ERROR TMKER_PrjToScreen (
    TMKER\_DESC TMKernel,
    S\_TMKER\_Point *pptPrj,
    S\_TMKER\_Point *pptScreen
);

```

TMKernel - [in] описатель экземпляра детектора

pptPrj - [in] точка в проективной системе координат

pptScreen - [out] точка в экранной системе координат

Возвращаемое значение: идентификатор ошибки.

В функции производится пересчёт координат точки из проективной системы координат в экранную. В проективной системе координат определяются точки непосредственно на дороге. Преобразование может быть использовано для визуализации событий или другой служебной графики. Пример преобразования приведен на рисунке.

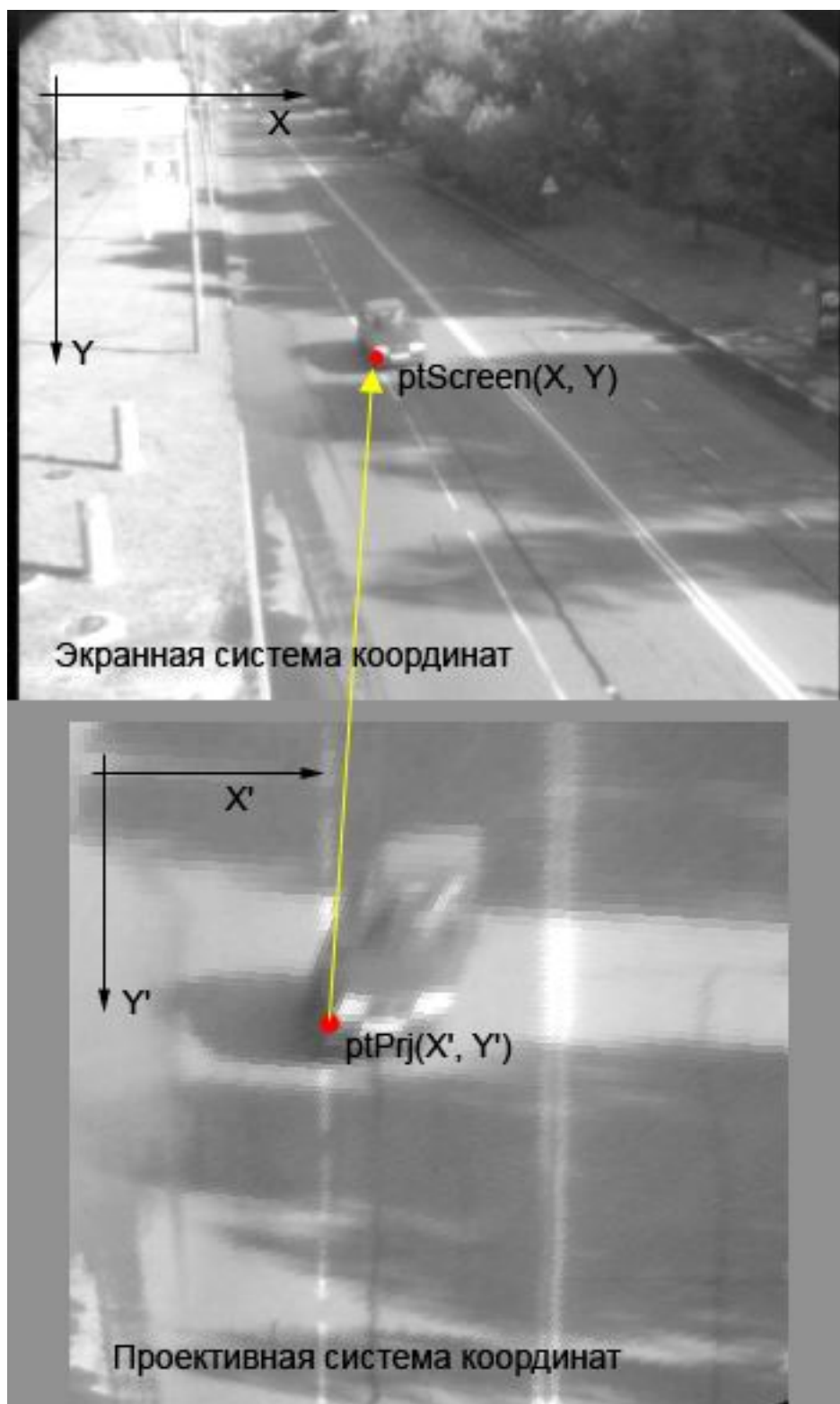


Рисунок 4.3 - Преобразование точки из проективной в экранную систему координат

Объявлено в TMKERNEL.h

4.2.8. TMKER_Release

Освобождение выделенных ресурсов

```
void TMKER_PrjToScreen(
    TMKER\_DESC TMKernel
);
```

TMKernel - [in] описатель экземпляра детектора.

Возвращаемое значение: идентификатор ошибки.

Функция освобождает все ресурсы, ранее выделенные в функции [TMKER_Init](#).

Объявлено в TMKERNEL.h

4.2.9. TMKER_ScreenToPrj

Преобразование точки из экранной в проективную систему координат

```
TMKER\_ERROR TMKER_ScreenToPrj(
    TMKER\_DESC TMKernel,
    S\_TMKER\_Point *pptScreen,
    S\_TMKER\_Point *pptPrj
);
```

TMKernel - [in] описатель экземпляра детектора.

pptScreen - [in] точка в экранной системе координат.

pptPrj - [out] точка в проективной системе координат.

Возвращаемое значение: идентификатор ошибки.

В функции производится пересчёт координат точки из экранной системы координат в проективную. Функция может быть использована для визуализации событий на дорожной полосе или другой служебной графики. Производимое преобразование показано на рисунке.

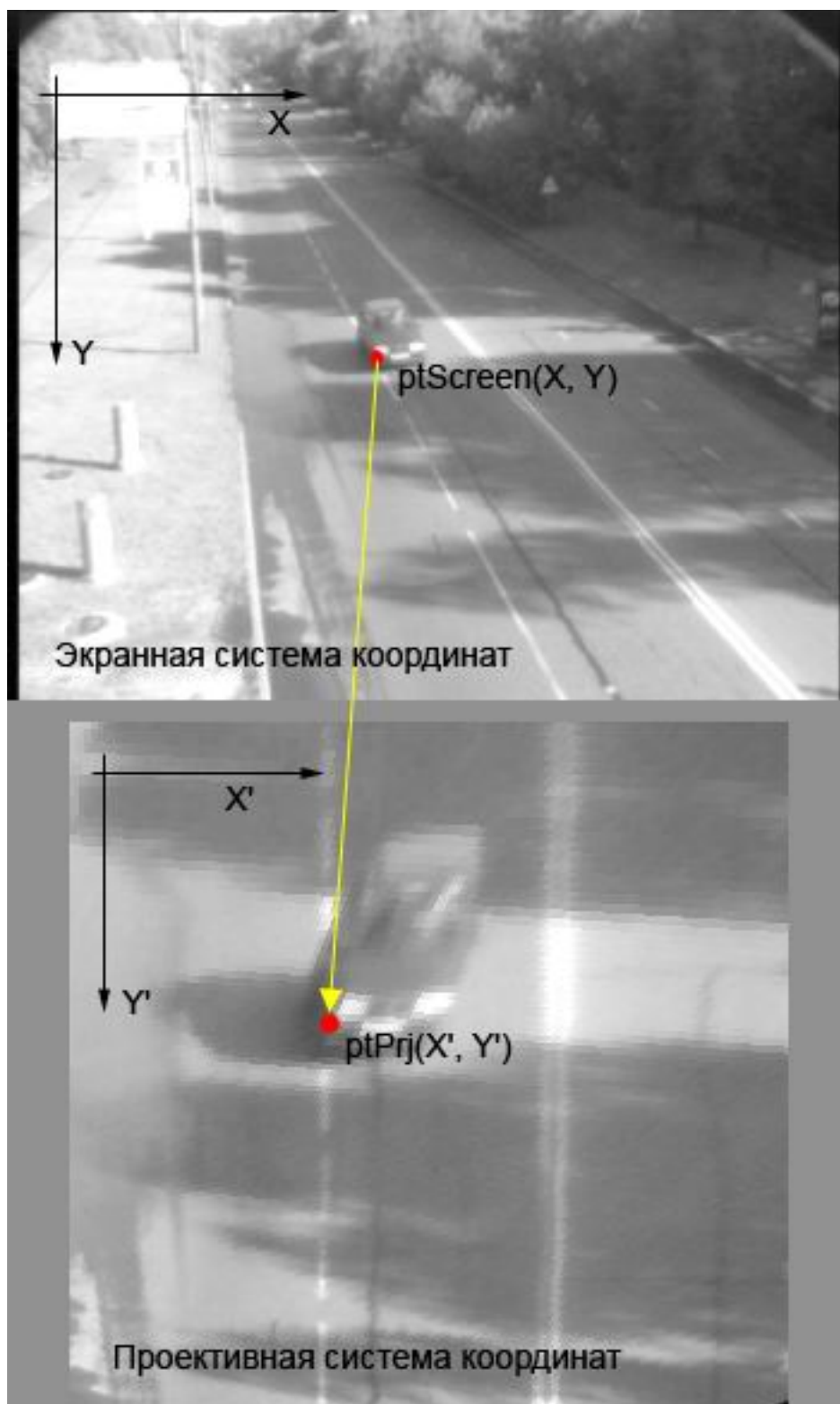


Рисунок 4.4 - Преобразование точки из экранных в проективные координаты

Объявлено в TMKERNEL.h

5. Описание AFFIXWIZARD API

5.1. Общие сведения

Мастер разметки является динамически подключаемой библиотекой и предоставляет функции для разметки детектора транспорта в интерактивном режиме, загрузки и сохранения разметки. После вызова функции разметки на экране в центре появляется диалоговое окно с набором страниц свойств.

Первая страница мастера разметки предназначена для "привязки" камеры к наблюдаемой сцене. Привязку можно выполнить либо задав внутренние параметры камеры и высоту её установки, либо задав опорные точки для определения ориентации камеры относительно дорожного полотна.

Для ввода внутренних параметров камеры необходимо включить радио кнопку "Задать параметры камеры" и ввести соответствующие значения (рис. [5.1](#)).



Рисунок 5.1 - Задание параметров камеры

Если же параметры камеры не известны, то необходимо задать опорные точки. Для этого нужно включить радио кнопку "Задать опорные точки" (рис. [5.2](#)).

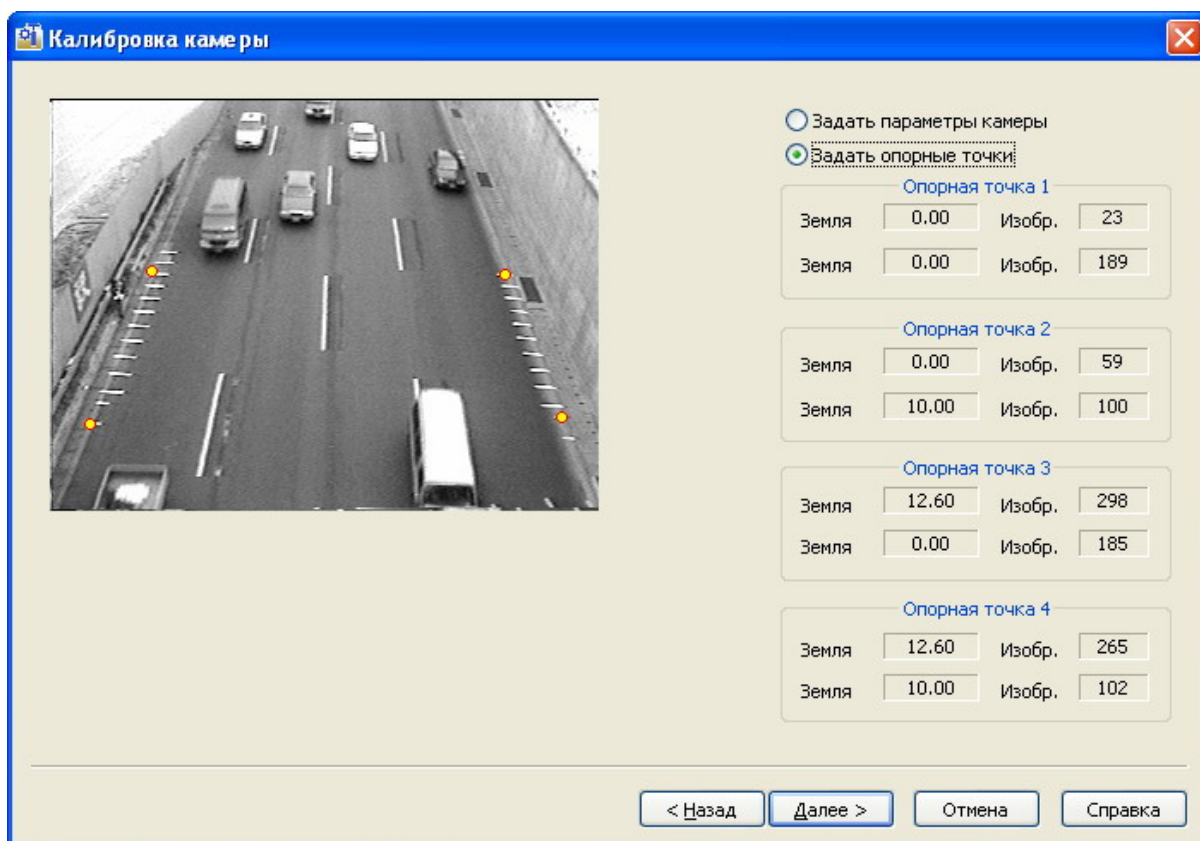


Рисунок 5.2 - Задание опорных точек

Необходимо наличие в поле зрения камеры четырех опорных точек. Опорная точка – малоразмерный объект, хорошо поддающийся идентификации на изображении. Опорными точками могут служить, например, метки, нанесенные краской на проезжей части. Необходимо измерить и записать координаты всех опорных точек в плоской декартовой системе координат, находящейся в плоскости дороги, с произвольно выбранным началом отсчета. Ось Y (ордината) направлена вдоль дороги, ось X (абцисса) - перпендикулярна дороге. Примеры выбора системы координат показаны на рисунке [5.3](#) (a, b - правильно, c - не правильно). На одной прямой не должны находиться более двух опорных точек. Опорные точки должны находиться на максимальном (по возможности) расстоянии друг от друга, но при этом быть в поле зрения камеры. Чем больше удалены опорные точки, тем точнее привязка.

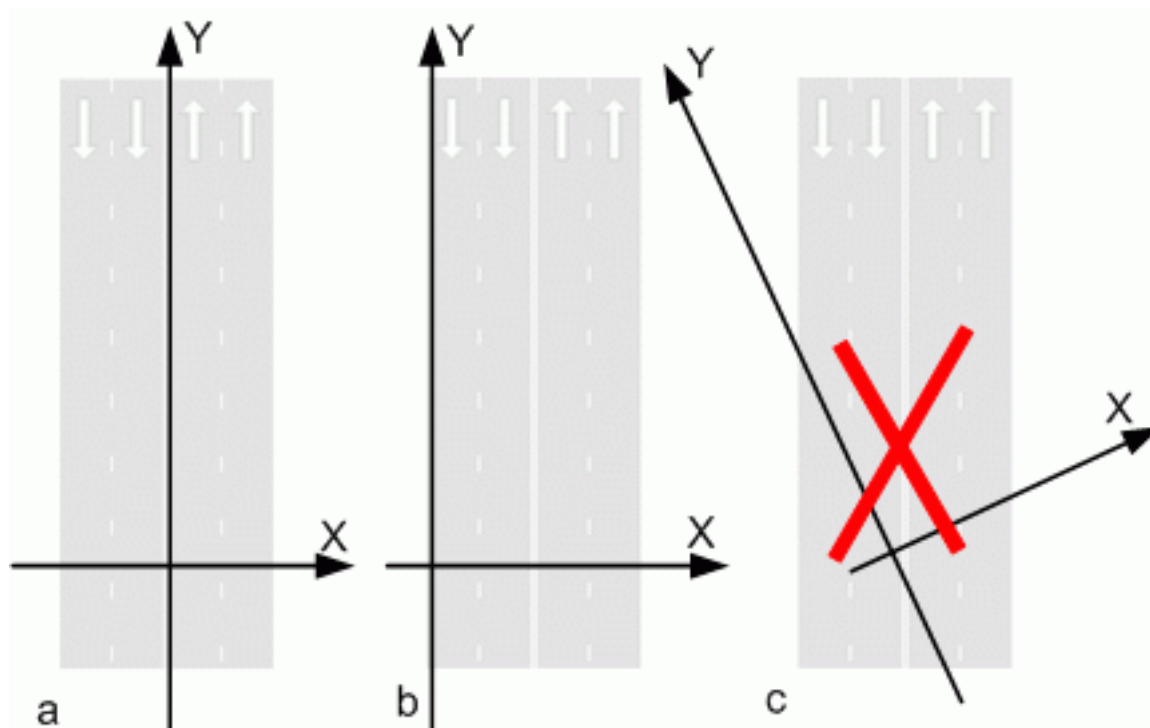


Рисунок 5.3 - Примеры выбора системы координат

Для определения опорной точки необходимо навести курсор на изображение опорной точки на экране и нажать левую кнопку мыши. В открывшемся окне с увеличенным фрагментом изображения нужно как можно точнее поставить маркер на изображение опорной точки (ее центр) и ввести ее координаты в метрах в системе отсчета, связанной с землей. Результат разметки можно видеть на правой панели окна. Для удаления точки нужно подвести курсор к её окрестности и, удерживая клавишу SHIFT, нажать левую кнопку мыши. Необходимо определить все четыре опорные точки.

На второй странице мастера задаются границы рабочей зоны (границы участка дороги, на котором будет осуществляться мониторинг транспортного потока), количество полос движения и направление движения по каждой из полос. Границы рабочей зоны задаются на изображении с помощью отрезков линий, направленных вдоль дороги. Хорошим ориентиром для задания границ рабочей зоны могут служить края дороги или линии дорожной разметки.



Рисунок 5.4 - Разметка рабочей области

Для задания границы необходимо выбрать на изображении начальную точку границы и нажать левую кнопку мыши. При перемещении курсора на экране будет отрисовываться линия. Подведите курсор к конечной точке и ещё раз нажмите левую кнопку мыши. Получившийся отрезок линии должен лежать строго вдоль края дороги либо вдоль разделительной линии дорожной разметки (если детектор размечается для анализа отдельных полос движения). Длина отрезка не имеет значения, но он не должен быть слишком коротким. Необходимо отметить левую и правую границы. Для удаления границы подведите курсор к одной из её крайних точек и, удерживая клавишу SHIFT, нажмите левую кнопку мыши.

Количество полос движения в рабочей области задаётся с помощью выпадающего списка *"Количество полос в зоне"*.

С помощью радио кнопок нужно задать правильное направление движения ТС для каждой из размеченных полос движения.

Результат произведённой разметки можно увидеть на третьей странице мастера разметки (рис. [5.5](#)). Результат работы может быть представлен также в виде проекции, то есть как бы вид сверху на рабочую зону (рис. [5.6](#)).

Если калибровка камеры была произведена по внутренним параметрам камеры, то на этой странице можно компенсировать возможный крен камеры (угол поворота вокруг продольной оси камеры). Для этого уточните разметку с помощью слайдера "Крен". Угол крена отображается в градусах.

Если калибровка произведена правильно, то отрисовываемая зона будет "лежать" на дороге, а продольные линии дорожной разметки на проективном изображении будут вертикальными.

На этой странице можно также изменить ширину каждой из полос движения. Для этого подведите курсор мыши к зелёной линии-разделителю полос. После изменения курсора на двунаправленную стрелку нажмите левую кнопку мыши. Удерживая кнопку, передвиньте линию-разделитель. Изменить ширину полос можно как на реальном, так и на проективном изображении.

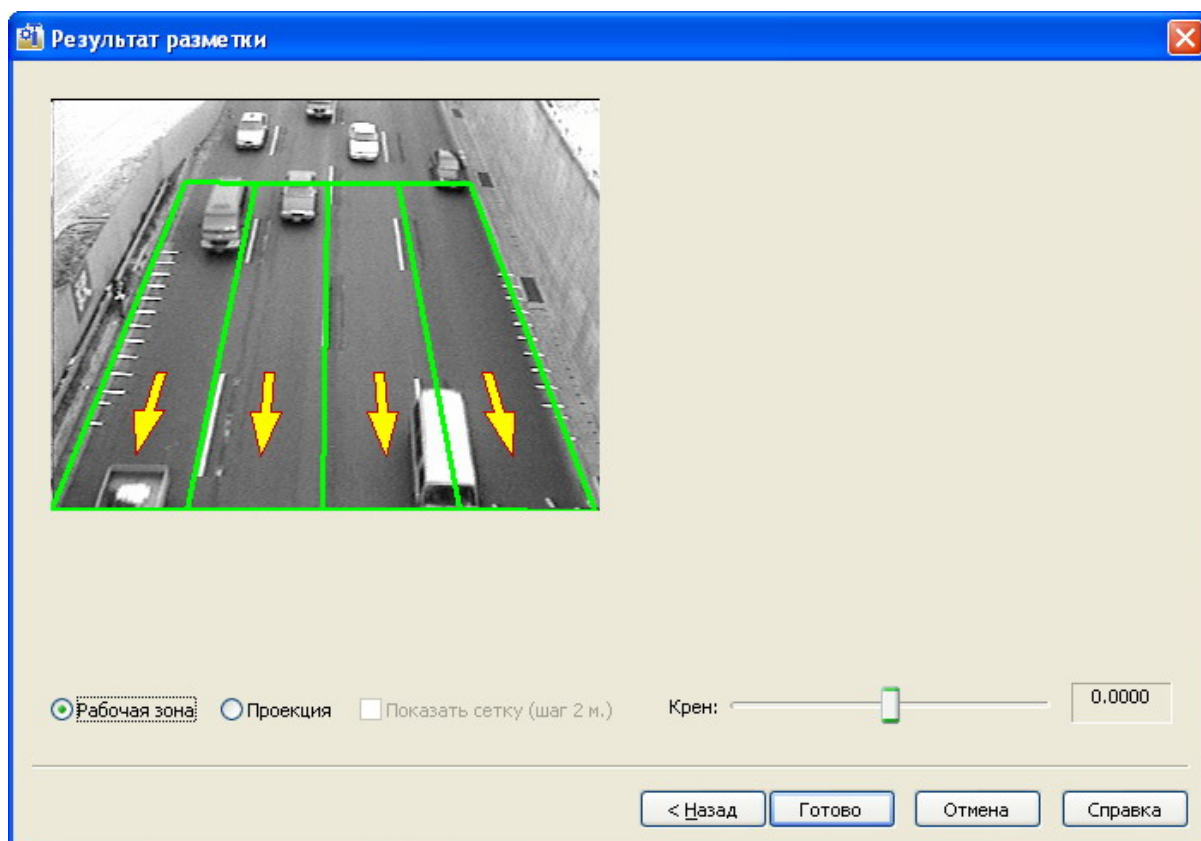


Рисунок 5.5 - Отображение результата разметки (реальный вид)

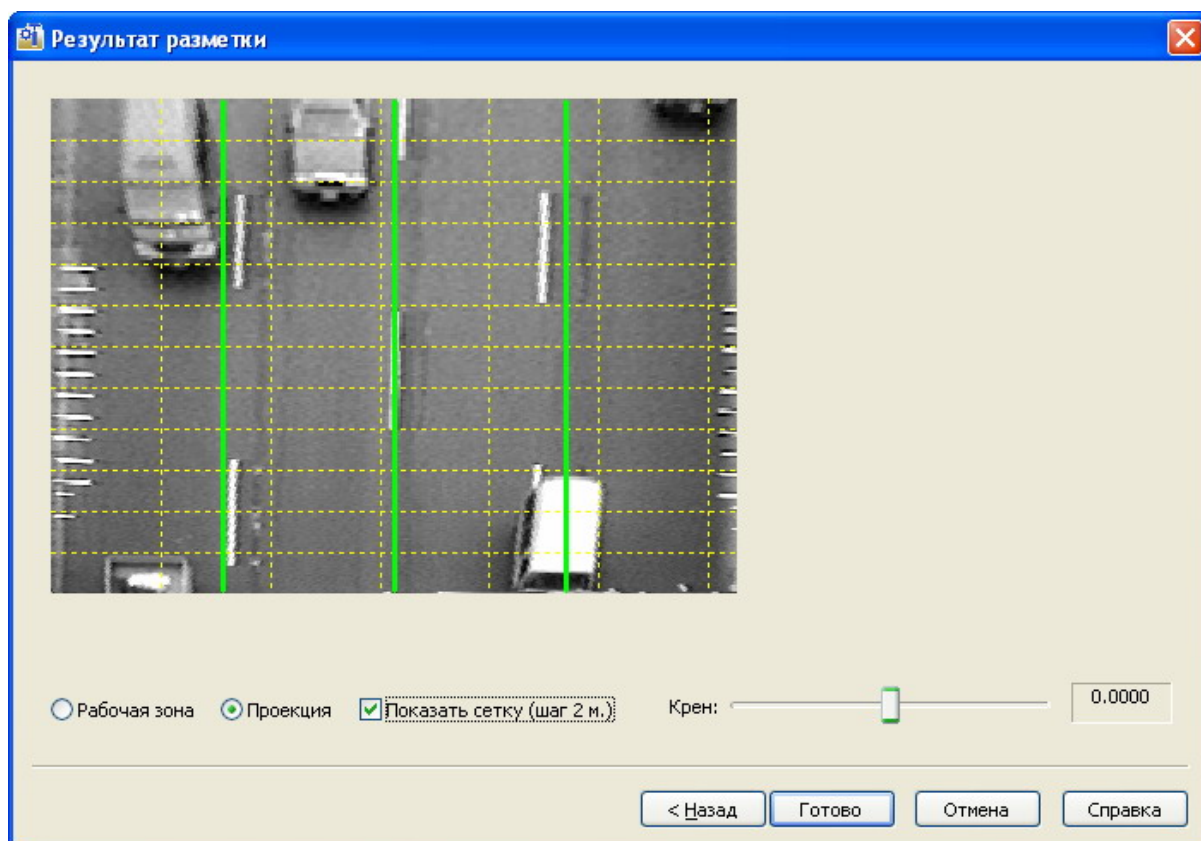


Рисунок 5.6 - Отображение результата разметки (проекция)

5.2. Структуры

5.2.1. S_AW_CamInternalParams

Внутренние параметры и высота установки камеры

```
typedef struct tagS_AW_CamInternalParams
{
    DWORD dwEffectivePixelsWidth;
    DWORD dwEffectivePixelsHeight;
}
```

```
double dblCellWidth;
double dblCellHeight;
double dblFocus;
double dblCameraHeight;
} S_AW_CamInternalParams;
```

dwEffectivePixelsWidth - Количество эффективных пикселей в строке матрицы камеры (ширина в пикселях).

dwEffectivePixelsHeight - Количество эффективных строк в матрице камеры (высота в пикселях).

dblCellWidth - Ширина пикселя в микрометрах.

dblCellHeight - Высота пикселя в микрометрах.

dblFocus - Фокусное расстояние объектива в миллиметрах.

dblCameraHeight - Высота установки камеры в метрах.

Объявлено в AFFIXWIZARD.h

5.3. Функции

5.3.1. AW_LoadFromFile

Загрузка разметки из файла

```
BOOL AW_LoadFromFile(
    LPCTSTR szFileName,
    S_TMKER_Affixment *pAffix,
    S_AW_CamInternalParams *pCIPParams
);
```

szFileName - [in] имя файла разметки

pAffix - [out] разметка

pCIPParams - [out] внутренние параметры и высота установки камеры

Возвращаемое значение: TRUE - нет ошибок, FALSE - ошибка.

Разметка, внутренние параметры и высота установки камеры загружаются из текстового файла в формате XML с кодировкой UTF-8.

Объявлено в AFFIXWIZARD.h

5.3.2. AW_SaveToFile

Сохранение разметки в файле

```
BOOL AW_SaveToFile(
    LPCTSTR szFileName,
    S_TMKER_Affixment *pAffix,
```

```
S\_AW\_CamInternalParams *pCIParams  
) ;
```

szFileName - [in] имя файла разметки

pAffix - [in] разметка

pCIParams - [in] внутренние параметры и высота установки камеры

Возвращаемое значение: TRUE - нет ошибок, FALSE - ошибка.

Разметка, внутренние параметры и высота установки камеры сохраняются в текстовом файле в формате XML с кодировкой UTF-8.

Объявлено в AFFIXWIZARD.h

5.3.3. AW_Show

Вызов мастера разметки

```
BOOL AW_Show(  
    HWND hwndOwner,  
    BYTE *pbImage,  
    DWORD dwImageSize,  
    S\_TMKER\_Affixment *pAffix,  
    S\_AW\_CamInternalParams *pCIParams  
) ;
```

hwndOwner - [in] окно-собственник мастера разметки. Параметр может принимать значение NULL

pbImage - [in] буфер с изображением

dwImageSize - [in] размер изображения в байтах

pAffix - [in][out] разметка

pCIParams - [in][out] внутренние параметры и высота установки камеры

Возвращаемое значение: TRUE - нет ошибок, FALSE - ошибка.

При отсутствии окна-собственника у мастера разметки, параметр *hwndOwner* можно установить в NULL.

Для корректного отображения кадра в мастере разметки рекомендуется передавать кадры шириной не более 384 пикс. и высотой не более 300 пикс.

Буфер *pbImage* должен содержать изображение в том формате, в котором это изображение записывается в файл, то есть считанные из файла данные изображения можно без изменения передавать в функцию. Поддерживаются изображения следующих форматов:

- BMP
- GIF
- JPEG

- PNG
- TIFF

Параметры `pAffix` и `pCIParams` являются как входными параметрами, так и результатом работы функции. Мастер разметки начнёт работу со входными параметрами и в этих же структурах сохранит все изменения.

Объявлено в `AFFIXWIZARD.h`

6. Описание API для коррекции разметки

6.1. Общие сведения

Коррекция разметки производится для уточнения результата калибровки видеокамеры. Для коррекции необходимо знать [внутренние параметры камеры и высоту её установки](#).

С помощью [мастера разметки](#) калибровку видеокамеры можно сделать двумя способами:

1. Задать опорные точки и границы рабочей зоны.
2. Задать внутренние параметры камеры, высоту её установки и границы рабочей зоны.

После калибровки способом 1 проводить коррекцию разметки нет необходимости.

При втором способе калибровки особенно важна точность задания линий границ рабочей зоны. Иногда - например, когда плохо просматриваются края дорожного покрытия - точно указать границы рабочей зоны затруднительно. Возможные неточности при задании этих границ могут увеличить погрешности измеряемых TMKernel характеристик. Суть коррекции разметки состоит в уточнении границ рабочей зоны.

Рассмотрим коррекцию разметки на примере.

На рисунке [6.1](#) показана исходная разметка, в которой видеокамера была откалибрована вторым способом, и где границы рабочей зоны были заданы вручную. Можно заметить, что линии границ зоны заданы неточно (левая часть картинки) и, в результате, на изображении справа видно, что рабочая зона повернута несколько вправо. Такая разметка может привести к увеличению погрешностей классификации, измерений скорости ТС и расстояний.

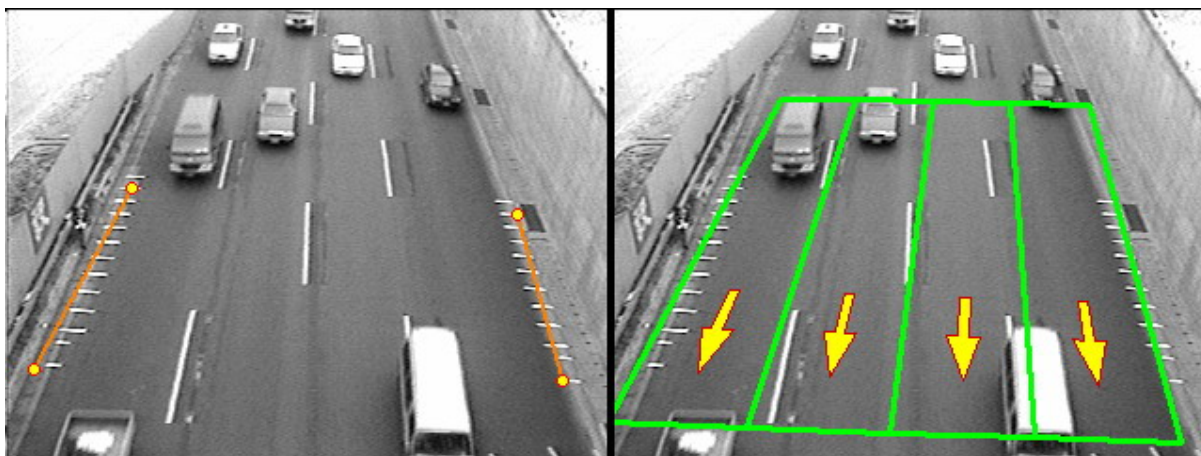


Рисунок 6.1 - Границы рабочей зоны до коррекции

После проведения автоматической коррекции (рис. 6.2) видно, что границы рабочей зоны выровнялись, и разметка правильно "ложится" на дорогу.



Рисунок 6.2 - Границы рабочей зоны после коррекции

Коррекция разметки производится по результатам обработки и анализа последовательности изображений. Для правильной работы модуля необходимо соблюдение следующих условий:

1. Видеокамера установлена в соответствии с [рекомендациями по установке камеры](#).
2. На дороге хорошо просматриваются линии разметки полос движения.

3. В области видимости дорога прямая и не имеет пересечений с другими дорогами.
4. Линии разметки полос движения прямые и параллельны направлению дороги.
5. Автомобильный поток не слишком плотный и отсутствуют дорожно-транспортные пробки.
6. Видеокадры обрабатываются последовательно и без пропусков.

Важно

При несоблюдении указанных условий, коррекция может выдать неправильный результат. В любом случае, скорректированную разметку следует использовать только после визуального просмотра результата. Используйте мастер разметки для визуализации скорректированной разметки.

В приведённом ниже листинге показан пример использования API для коррекции разметки.

```

1. #include <windows.h>
2. #include <tchar.h>
3. #include <stdio.h>
4. #include "AFFIXWIZARD.h"
5. #include "CorrectAffixment.h"

6. extern BOOL GetFrame(BYTE *pbFrame);
7. extern void ConvertKerToCA(S_TMKER_Affixment *pAffix,
8.     S_AW_CamInternalParams *pCIP, S_CA_Camera *pCamera,
9.     S_CA_Edges *pImgEdges);
10. extern void ConvertCAToKer(S_CA_Edges *pImgEdges,
11.     S_CA_Edges *pEarthEdges, S_TMKER_Affixment *pAffix);
12. static const int FRAME_WIDTH = 352;
13. static const int FRAME_HEIGHT = 288;

14. void main()
15. {
16.     S_TMKER_Affixment Affix;
17.     S_AW_CamInternalParams CIPParams;
18.     BYTE *pbFrame = NULL;
19.     CA_HANDLE hHS = NULL;
20.     S_CA_Camera Camera;
21.     S_CA_Edges SrcImgEdges;
22.     S_CA_Edges DstImgEdges;
23.     S_CA_Edges DstEarthEdges;
24.     int nFrames;
25.     int i;

26.     try{
27.         if(!AW_LoadFromFile(_T("OldAffix.xml"), &Affix,
28.             &CIPParams)){

```

```

29.         throw _T("Fail to load affixment");
30.     }
31.     pbFrame = new BYTE[FRAME_WIDTH * FRAME_HEIGHT];
32.     if(!pbFrame){
33.         throw _T("Fail to allocate memory");
34.     }
35.     if(CA_Create(&hHS)){
36.         throw _T("Fail to create corrector");
37.     }
38.     if(CA_Init(hHS, FRAME_WIDTH, FRAME_HEIGHT)){
39.         throw _T("Fail to init corrector");
40.     }
41.     ConvertKerToCA(&Affix, &CIParams, &Camera,
42.         &SrcImgEdges);
43.     if(CA_SetCameraParams(hHS, &Camera, &SrcImgEdges)){
44.         throw _T("Fail to set camera parameters");
45.     }
46.     nFrames = CA_GetFramesForBkgr(hHS);
47.     for(i=0; i<nFrames; i++){
48.         if(!GetFrame(pbFrame)){
49.             throw _T("Fail to get frame from AVI");
50.         }
51.         if(CA_SetImage(hHS, pbFrame)){
52.             throw _T("Fail to copy frame in internal beffer");
53.         }
54.     }
55.     for(i=0; i<nFrames; i++){
56.         if(CA_Operate(hHS)){
57.             throw _T("Fail to make correction");
58.         }
59.     }
60.     if(CA_CorrectAffixment(hHS, &DstImgEdges,
61.         &DstEarthEdges, 10)){
62.         throw _T("No corrected affixment");
63.     }
64.     ConvertCAToKer(&DstImgEdges, &DstEarthEdges, &Affix);
65.     if(!AW_SaveToFile(_T("NewAffix.xml"), &Affix,
66.         &CIParams)){
67.         throw _T("Fail to save affixment");
68.     }
69. }
70. catch(TCHAR *szError){
71.     if(szError) {
72.         _tprintf(_T("Error: %s\n"), szError);
73.     }
74. }
75. delete [] pbFrame;

```

```

76.    CA_Release(hHS);
77.    CA_Destroy(hHS);
78. }

```

1 - 5: подключение необходимых заголовочных файлов.

6: экспортируется функция для получения одного видео кадра. Функция принимает на вход буфер для кадра. Предполагается, что каждый вызов GetFrame() заполняет буфер pbFrame новым кадром из последовательности видео кадров. Возвращает булево значение: TRUE - кадр успешно получен, FALSE - кадр отсутствует. Размер кадра задаётся в строках 2 и 3. Для работы с TMKernel требуется кадр с одним байтом на пиксель (256 градаций серого). В буфере pbFrame кадр размещается в правильном порядке, то есть первая строка кадра располагается в начале буфера. Функция GetFrame создается пользователем и в листинге не приводится.

7 - 9: экспортируется функция для конвертирования структур, описанных в TMKERNEL.h и AFFIXWIZARD.h в структуры, описанные в CorrectAffixment.h. Функция ConvertKerToCA создается пользователем и в листинге не приводится.

10, 11: экспортируется функция для конвертирования структур, описанных в CorrectAffixment.h в структуры, описанные в TMKERNEL.h и AFFIXWIZARD.h. Функция ConvertCAToKer создается пользователем и в листинге не приводится.

12, 13: задается ширина и высота видеокадров.

16 - 25: объявление переменных.

26: вход в блок try-catch.

27 - 30: загрузка разметки и внутренних параметров камеры из файла.

31 - 34: выделение памяти для хранения видеокадра.

35 - 37: получение описателя экземпляра объекта для коррекции разметки.

35 - 37: Get instance descriptor for Affixment Correction module (corrector).

38 - 40: инициализация.

41 - 45: передача параметров камеры и координаты краёв дороги.

41 - 45: Supply corrector by camera parameters and boundaries information.

46: получение количества кадров для выполнения коррекции.

47 - 54: цикл загрузки изображений во внутренний буфер модуля.

48 - 50: получение одного кадра.

51 - 53: загрузка изображения во внутренний буфер модуля.

55 - 59: цикл обработки.

56 - 58: покадровая обработка*.

60 - 63: получение результата коррекции разметки.

64 - 68: сохранение скорректированной разметки в файле.

71 - 73: вывод ошибки.

75 - 77: освобождение ресурсов.

Циклы загрузки изображений во внутреннюю память модуля и цикл обработки изображений можно объединить:

```
for(i=0; i<nFrames; i++){
    if(!GetFrame(pbFrame)){
        throw _T("Fail to get frame from AVI");
    }
    if(CA_SetImage(hHS, pbFrame)){
        throw _T("Fail to copy frame in internal beffer");
    }
    if(CA_Operate(hHS)){
        throw _T("Fail to make correction");
    }
}
```

Раздельная загрузка и обработка позволяет копировать кадры от видеоисточника "на ходу", так как перенос кадра во внутренний буфер модуля происходит быстро и без пропусков кадров. Обработка же кадров занимает значительно большее время, поэтому, если обработку выполнять "на ходу" - сразу после копирования - возможны пропуски кадров.

6.1.1. Итерационная коррекция разметки

Коррекция разметки может быть выполнена за несколько последовательных итераций. Такой подход позволяет произвести коррекцию разметки даже если соблюдены не все условия для выполнения правильной коррекции. Например, если в момент первой итерации автомобильный поток был слишком плотным (что снижает качество или делает невозможной коррекцию), то можно выполнить дополнительную коррекцию по новой последовательности видеокадров.

Дело в том, что после обработки последовательности и получения скорректированной разметки внутренние данные сохраняются и будут учитываться при коррекции по новой последовательности видеокадров. Таким образом, повторная, а также все последующие итерации будут уточнять полученные в предыдущих итерациях данные, необходимые для разметки детектора.

Выполнение любой итерации можно прервать на этапе накопления и обработки кадров, вызвав после этого функцию [CA_Reset\(\)](#). При этом не происходит сброса внутренних данных и сохраняется возможность выполнить очередную итерацию в другое время.

Полученные в результате обработки последовательностей кадров внутренние данные сбрасываются только после новой [инициализации](#).

6.2. Структуры и идентификаторы

6.2.1. CA_HANDLE

Описатель экземпляра объекта для коррекции разметки

```
typedef int CA_HANDLE;
```

Программный модуль позволяет создать несколько экземпляров объектов коррекции в одном процессе. Каждый экземпляр работает независимо друг от друга. Для идентификации экземпляра необходимо передавать описатель во все функции API, которые требуют этот описатель.

Объявлено в CorrectAffixment.h

6.2.2. S_CA_Camera

Внутренние параметры камеры и высота её установки

```
typedef struct tagS_CA_Camera
{
    double dblCamHeight;
    double dblFocus;
    int nEPWidth;
    int nEPHeight;
    double dblCellWidth;
    double dblCellHeight;
} S_CA_Camera;
```

dblCamHeight - Высота установки камеры в метрах

dblFocus - Фокусное расстояние объектива в миллиметрах

nEPWidth - Количество эффективных пикселей в строке матрицы камеры (ширина в пикселях)

nEPHeight - Количество эффективных строк в матрице камеры (высота в пикселях)

dblCellWidth - Ширина пикселя в микрометрах

dblCellHeight - Высота пикселя в микрометрах

Объявлено в CorrectAffixment.h

6.2.3. S_CA_Edges

Границы рабочей области детектора

```
typedef struct tagS_CA_Edges
{
    double dlbLeftTopX;
    double dlbLeftTopY;
    double dlbLeftBottomX;
    double dlbLeftBottomY;
    double dlbRightTopX;
    double dlbRightTopY;
    double dlbRightBottomX;
    double dlbRightBottomY;
} S_CA_Edges;
```

dlbLeftTopX - X координата верхней точки отрезка, задающего левую границу области.

dlbLeftTopY - Y координата верхней точки отрезка, задающего левую границу области.

dlbLeftBottomX - X координата нижней точки отрезка, задающего левую границу области.

dlbLeftBottomY - Y координата нижней точки отрезка, задающего левую границу области.

dlbRightTopX - X координата верхней точки отрезка, задающего правую границу области.

dlbRightTopY - Y координата верхней точки отрезка, задающего правую границу области.

dlbRightBottomX - X координата нижней точки отрезка, задающего правую границу области.

dlbRightBottomY - Y координата нижней точки отрезка, задающего правую границу области.

Все значения определяются в либо в экранных координатах (на изображении) в пикселях, либо в земных координатах в метрах, в зависимости от целевого использования структуры.

Объявлено в CorrectAffixment.h

6.3. Функции

6.3.1. CA_CorrectAffixment

Коррекция разметки

```
int CA_CorrectAffixment(
    CA_HANDLE hCA
    S_CA_Edges *pDstImgEdges,
    S_CA_Edges *pDstEarthEdges
);
```

hCA - [in] описатель объекта для коррекции.

pDstImgEdges - [out] линии, задающие скорректированные границы рабочей области на изображении (пиксели)

pDstEarthEdges - [out] линии, задающие скорректированные границы рабочей области на земле (метры)

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Функция производит коррекцию координат отрезков линий, которые задают границы рабочей области. Перед вызовом этой функции необходимо произвести инициализацию ([CA_Init\(\)](#)), установить параметры камеры ([CA_SetCameraParams\(\)](#)) и произвести обработку последовательности видеок кадров с помощью вызовов [CA_SetImage\(\)](#) и [CA_Operate\(\)](#). Количество кадров в последовательности можно определить вызовом [CA_GetFramesForBkgr\(\)](#).

Объявлено в `CorrectAffixment.h`

6.3.2. CA_Create

Создание экземпляра объекта для коррекции разметки

```
int CA_Create(CA\_HANDLE *phCA);
```

phCA - [out] указатель на описатель объекта для коррекции

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Описатель необходим для идентификации экземпляра объекта коррекции разметки и используется как параметр для остальных функций API.

Для уничтожения созданного экземпляра необходимо вызвать [CA_Destroy\(\)](#).

Объявлено в `CorrectAffixment.h`

6.3.3. CA_Destroy

```
int CA_Destroy(  
    CA\_HANDLE hCA  
);
```

hCA - [in] описатель удаляемого экземпляра объекта для коррекции

Возвращаемое значение: всегда 0.

Уничтожает экземпляр объекта, который ранее был создан вызовом [CA_Create\(\)](#).

После вызова `CA_Destroy` не допускается дальнейшее использование этого описателя. Рекомендуется после удаления присвоить описателю нулевое значение.

Объявлено в CorrectAffixment.h

6.3.4. CA_GetFramesForBkgr

Возвращает количество кадров в последовательности, используемой для коррекции разметки

```
int CA_GetFramesForBkgr(CA\_HANDLE hCA);
```

hCA - [in] описатель объекта для коррекции разметки.

Возвращаемое значение: количество последовательных кадров, необходимых для коррекции разметки.

Объявлено в CorrectAffixment.h

6.3.5. CA_Init

Инициализация экземпляра объекта для коррекции разметки

```
int CA_Init(  
    CA\_HANDLE hCA,  
    int nImageWidth,  
    int nImageHeight  
);
```

hCA - [in] описатель объекта для коррекции разметки

nImageWidth - [in] ширина обрабатываемых изображений (пиксели)

nImageHeight - [in] высота обрабатываемых изображений (пиксели)

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Функция инициализации должна быть вызвана до использования остальных функций API, кроме функций [CA_Create\(\)](#) и [CA_Destroy\(\)](#).

Перед повторными вызовами [CA_Init](#) необходимо вызывать функцию [CA_Release\(\)](#).

Объявлено в CorrectAffixment.h

6.3.6. CA_Operate

Обработка одного видеокадра

```
int CA_Operate(  
    CA\_HANDLE hCA  
);
```

hCA - [in] описатель объекта для коррекции разметки

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Функция обрабатывает один видеокادر из внутренней памяти модуля. Перед вызовом этой функции необходимо скопировать видеокادر во внутренний буфер с помощью вызова [CA_SetImage\(\)](#). Для коррекции разметки необходимо обработать все N кадров, где N - возвращаемое функцией [CA_GetFramesForBkgr\(\)](#) значение. Возможны два способа обработки последовательности видеокладов:

1. Вызывать `CA_Operate()` сразу после копирования кадра:

```
nFrames = CA_GetFramesForBkgr();
for(i=0; i<nFrames; i++){
    CA_SetFrame(hCA, pbFrame);
    CA_Operate(hCA);
}
```

2. Сначала скопировать всю последовательность видеокладов и потом обработать:

```
nFrames = CA_GetFramesForBkgr();
for(i=0; i<nFrames; i++){
    CA_SetFrame(hCA, pbFrame);
}
for(i=0; i<nFrames; i++){
    CA_Operate(hCA);
}
```

Если предполагается обработка кадров от видеоисточника, то следует выбрать второй способ, т. к. использование первого способа может привести к пропуску кадров. Это обусловлено тем, что время обработки одного кадра (вызов `CA_Operate`) может существенно превышать межкадровый интервал, что приведёт к пропускам кадров. Во втором способе в первом цикле происходит только копирование последовательности видеокладов во внутренний буфер, а обработка происходит во втором цикле, когда вся последовательность уже находится во внутренней памяти модуля.

После обработки всех кадров последовательности можно вызвать [функцию коррекции разметки](#).

Объявлено в `CorrectAffixment.h`

6.3.7. CA_Release

Освобождение выделенных при инициализации ресурсов

```
int CA_Release(
    CA\_HANDLE hCA
);
```

hCA - [in] описатель объекта для коррекции разметки.

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Объявлено в CorrectAffixment.h

6.3.8. CA_Reset

Сброс внутренних данных

```
int CA_Reset(
    CA_HANDLE hCA
);
```

hCA - [in] описатель объекта для коррекции разметки.

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Функция сбрасывает счётчик обработанных видеок кадров и полученные промежуточные внутренние результаты обработки последовательности видеок кадров для прерывания цикла обработки кадров и начала работы с новой последовательностью кадров.

Объявлено в CorrectAffixment.h

6.3.9. CA_SetCameraParams

Установка параметров камеры

```
int CA_SetCameraParams(
    CA_HANDLE hCA
    S_CA_Camera *pCamera,
    S_CA_Edges *pImgEdges
);
```

hCA - [in] описатель объекта для коррекции.

pCamera - [in] внутренние параметры камеры и её высота.

pSrcImgEdges - [in] линии, задающие границы рабочей области на изображении (пиксели).

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Объявлено в CorrectAffixment.h

6.3.10. CA_SetImage

Копирование одного видеок кадра во внутренний буфер модуля

```
int CA_SetImage(
```

```
CA\_HANDLE hCA,  
unsigned char *ucImage  
);
```

hCA - [in] описатель объекта для коррекции разметки.

ucImage - [in] передаваемый видеокадр.

hCA - [in] corrector instance descriptor

ucImage - [in] pointer to frame

Возвращаемое значение: 0 - нет ошибок, другое значение - ошибка.

Ширина и высота передаваемого кадра должны быть равны параметрам *nImageWidth* и *nImageHeight* структуры, которые были переданы в [функцию инициализации](#). Глубина цвета - один байт на пиксель (256 градаций серого). В буфере *ucImage* кадр должен размещаться в правильном порядке, то есть первая строка кадра располагается в начале буфера.

Объявлено в *CorrectAffixment.h*

7. Работа в демонстрационном режиме

При отсутствии ЭК TMKernel автоматически переходит в демонстрационный режим работы. В этом режиме всегда используется одна, "зашитая" в программу разметка детектора. Эта разметка соответствует демонстрационному AVI файлу (Demo.avi) из дистрибутивного пакета.

Если ЭК отсутствует, то функция [TMKER_Init\(\)](#) возвращает ошибку [TMKER_ER_DONGLE](#). При этом внутренние параметры TMKernel инициализируются некоторыми предопределёнными, демонстрационными параметрами. То есть, после вызова [TMKER_Init\(\)](#) с ошибкой [TMKER_ER_DONGLE](#) можно продолжить работу, однако детектор будет настроен на сцену, записанную в Demo.avi.

```
...
TMKER_ERROR Res;

Res = TMKER_Init(TMK, &Init);
if(TMKER_ER_DONGLE == Res){
    // Можно продолжить работу только в
    // демонстрационном режиме.
}
else if(TMKER_ER_OK != Res){
    // Ошибка.
}
...
```

8. Приложение

8.1. Измерение характеристик дорожного движения

8.1.1. Среднее значение и среднеквадратичное отклонение скорости

Среднее значение скорости на полосе движения за время T определяется

$$\bar{V} = \frac{\sum_{i=1}^n V_i}{n}, i \in [1..n]$$

как:

Здесь, n - количество ТС за время T , V_i - скорость отдельного ТС. При расчете средней скорости учитываются ТС, имеющие ненулевое значение длины и положительное значение скорости (возвращаются в структуре [S_TMKER_LaneEvent](#) после вызова функции [TMKER_ProcessFrame](#)).

Среднеквадратичное отклонение скорости за время T можно определить по формуле:

$$\sigma_v = \sqrt{\frac{\sum_{i=1}^n V_i^2 - \frac{\left(\sum_{i=1}^n V_i\right)^2}{n}}{n-1}}$$

Здесь, n - количество обнаруженных ТС за время T , V_i - скорость отдельного обнаруженного ТС.

Эта формула выводится из формулы подсчёта статистического среднеквадратичного

$$\sigma_v = \sqrt{\frac{\sum_{i=1}^n (V_i - \bar{V})^2}{n-1}}, i \in [1..n]$$

отклонения:

При вычислении среднеквадратичного отклонения с использованием `TMKernel` удобнее пользоваться первой формулой, так как здесь вычисления можно производить "на лету", накапливая сумму скоростей ТС и сумму квадратов скоростей ТС, не зная заранее среднее значение скорости. При вычислениях сумм следует учитывать возможные переполнения.

Средние параметры скорости можно вычислять как для всех обнаруженных ТС, так и по каждому из классов ТС отдельно (см. описание [TMKER_VEHICLE_CLASS](#)).

8.1.2. Занятость

Занятость полосы движения (О) за время Т можно определить, как процентное отношение времени, когда ТС находились в зоне измерения занятости (Т') к общему времени Т: $O = (T' / T) * 100\%$. Если вызовы [TMKER_ProcessFrame](#) происходили без пропусков кадров, то время Т' можно заменить количеством кадров, в которых поле nIsVehicleInDetectionArea структуры [S_TMKER_LaneEvent](#) имело ненулевое значение, а время Т можно заменить общим количеством кадров за время анализа. Описание зоны измерения занятости см. в описании структуры [S_TMKER_LaneEvent](#).

8.1.3. Обнаружение останова ТС и движения по встречной по

Для обнаружения останова ТС необходимо проанализировать поля nVehicleSpeed и nVehicleLength структуры [S_TMKER_LaneEvent](#) после вызова функции [TMKER_ProcessFrame](#). В случае останова ТС поле nVehicleSpeed будет иметь нулевое значение, а поле nVehicleLength ненулевое значение.

Движение по встречной полосе определяется как движение ТС в направлении, обратном заданному при разметке (см. описание структуры [S_TMKER_Affixment](#)). В этом случае ТС будет иметь отрицательную скорость (поле nVehicleSpeed структуры [S_TMKER_LaneEvent](#)). Абсолютное значение поля nVehicleSpeed - это скорость движущегося в обратном направлении ТС.

8.1.4. Обнаружение дорожно-транспортной пробки

Для обнаружения дорожно-транспортной пробки на полосе движения можно использовать следующие признаки:

- В течение определённого времени Т1 занятость полосы превышает заданную величину О.
- В течение определённого времени Т1 средняя скорость движения не превышает заданную величину V.

Дорожно-транспортная пробка фиксируется только тогда, когда присутствуют оба признака.

Для обнаружения окончания дорожно-транспортной пробки на полосе движения можно использовать следующие признаки:

- После обнаружения пробки в течение определённого времени Т2 занятость не превышает заданную величину О
- После обнаружения пробки в течение определённого времени Т2 средняя скорость движения превышает заданную величину V

Окончание дорожно-транспортной пробки фиксируется тогда, когда присутствует хотя бы один из признаков.

The traffic jam ending is fixed if either of these rules is satisfied.

Значения параметров T1, T2, O и V определяются в зависимости от требуемой "чувствительности" детектора и особенностей транспортного потока на данном участке дороги. Для большинства случаев можно использовать следующие значения:

- T1 = 15 секунд
- T2 = 20 секунд
- O = 30%
- V = 30 км/ч

Определение параметра "Занятость" описано в разделе ["Занятость"](#).

Определение средней скорости движения описано в разделе ["Среднее значение и среднеквадратичное отклонение скорости"](#).



НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

ЗАО НТЦ "Модуль"
А/Я 166, Москва, 125190, Россия
Тел: +7 (499) 152-9698
Факс: +7 (499) 152-4661
E-Mail: rusales@module.ru
WWW: <http://www.module.ru>

©ЗАО НТЦ "Модуль", 2008-2012

Все права сохранены.

Никакая часть информации, приведенная в данном документе, не может быть адаптирована или воспроизведена, кроме как согласно письменному разрешению владельцев авторских прав. ЗАО НТЦ "Модуль" оставляет за собой право производить изменения как в описании, так и в самом продукте без дополнительных уведомлений. ЗАО НТЦ "Модуль" не несет ответственности за любой ущерб, причиненный использованием информации в данном описании, ошибками или недосказанностью в описании, а также путем неправильного использования продукта.

Напечатано в России. Дата публикации: 02/07/2015