# Содержание

## Урок 1



## Урок 2



## Урок 3



## Урок 4

## Урок 5



## Урок 6



## Урок 7



## Урок 8

# Компьютерное зрение. Введение

УРОК 1

*В этом уроке изложены вопросы:*
*A. Компьютерное зрение пришло в школу*
*B. Задачи машинного зрения*
*C. Модульная парадигма Д. Марра обработки изображений*
*D. Предисловие*

## A. Компьютерное зрение пришло в школу

Компьютерное зрение стало популярным еще в 1960-х годах, но сейчас оно находится в особой точке своего развития. Компьютерное зрение пришло в школу. Сейчас появилась возможность создания полезных программ обработки изображений для проведения исследований в школе.

Реализация компьютерного зрения столь привлекательна для исследователей по той причине, что аппаратные возможности в данной области достигли такого уровня, что они уже во многом приближаются к «техническим характеристикам» зрения человека. Разрешение многих сенсоров для получения видеоинформации практически соответствует числу клеток сетчатки глаза человека, а возможности компьютеров и специальных процессоров близки к характеристикам «вычислительных мощностей» мозга. Процесс снятия изображения недеструктивен, достаточно прост и недорог на сегодняшний момент. И становится уже реально в школе не визуально, «на глазок» оценивать развитие исследуемого процесса, а дать более точную его оценку по цифровым изображениям.

У информационного содержания цифрового изображения нет какой-либо «причинной» или динамической модели формирования, оно не подвластно только каким-либо общим физическим законам и сложным математическим уравнениям. Оно представляет из себя бесконечное разнообразие яркостно-геометрических структур [1]. И на первый взгляд может показаться, что обработка таких «несложных объектов» не требует особых знаний и умений. Следует учесть, что компьютерное зрение – это пограничная область знаний. Для извлечения полезной информации надо уметь применять статистические методы, использовать модели, построенные с помощью геометрии, физики и теории обучения. В этой области знаний нет непререкаемых авторитетов, на которые можно сослаться, многие полезные идеи не имеют под собой теоретической основы, а некоторые теории бесполезны

на практике [2]. Пока нет единого математического формализма и единой общепризнанной методики разработки алгоритмов анализа изображений. Переживая период бурного роста, наука об обработке изображений все еще находится на одной из начальных стадий своего развития. Разработка и использование моделей, пригодных для эффективного решения задачи обнаружения соответствующих объектов на изображении, в значительной степени остается на грани науки и искусства и зачастую основывается на многолетнем опыте исследований частных задач [1].

Приведем предложенную в [1] классификацию понятий «зрение роботов» (robot vision), «компьютерное зрение» (computer vision), «обработка изображений» (image processing), «понимание изображений» (image understanding).

**Компьютерное зрение** представляет собой научную дисциплину, изучающую теорию и базовые алгоритмы анализа изображений и сцен.

**Машинное зрение** следует рассматривать как более комплексную и технологическую область научных и инженерных знаний, охватывающую все проблемы разработки практических систем: выбор схем освещения исследуемой сцены, выбор характеристик датчиков, их количества и геометрии расположения, вопросы калибровки и ориентирования, выбор или разработка оборудования для оцифровки и процессорной обработки, разработка собственно алгоритмов и их компьютерная реализация – то есть весь круг сопутствующих задач.

**Зрение роботов** предлагаем трактовать как более узкую область технологий машинного зрения, а именно ту их часть, которая обеспечивает функционирование систем машинного зрения в условиях жестких временных ограничений. К этому понятию, безусловно, относятся проблемы разработки основанных на изображениях информационных систем, входящих в состав систем управления сложными динамическими объектами (самолет, автомобиль, системы контроля технических и технологических процессов на производстве), так как необходимость формирования обратных связей по результатам обработки входных изображений в системах управления, очевидно, требует их быстрого анализа в режиме реального времени.

## B. Задачи машинного зрения

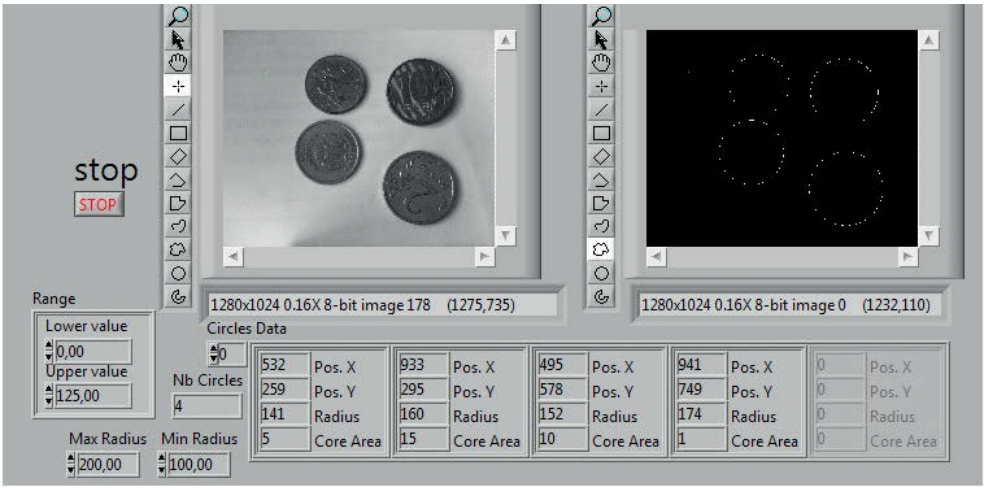
Машинное зрение имеет огромное число потенциальных областей применения, таких как промышленная инспекция и контроль качества, робототехника, навигация и транспортировка, дистанционное зондирование, медицина и биомеханика, инженерный труд, автоматизация проектирования, новые технологии обработки документов, биометрия и множество других. Мы будем использовать его для организации зрительной обратной связи

при работе управляемых устройств, манипуляторов или мобильных роботов в изменчивой среде.

Основные задачи машинного зрения могут быть сформулированы следующим образом:

1) обнаружение объектов и изменений в сцене наблюдения;
2) описание сцены и идентификация объектов;
3) слежение за объектами;
4) калибровка сенсоров, самоориентация и самопозиционирование;
5) реконструкция поверхностей и обнаружение трехмерных структур;
6) высокоточные измерения элементов сцены;
7) организация зрительной обратной связи при работе управляемых устройств, манипуляторов или мобильных роботов в изменчивой среде.

В данной книге мы рассмотрим только первую задачу. Но и даже ее решение позволит нам реализовать многие интересные проекты.




## C. Модульная парадигма Д. Марра обработки изображений

В алгоритмическом аспекте последовательность действий по обработке изображения принято рассматривать в согласии с так называемой модульной парадигмой, информационной теорией зрения Дэйвида Марра (он работал в Лаборатории искусственного интеллекта Массачусетского технологического института, США), получившей в настоящее время широкую известность [3]. Суть теории Д. Марра состоит в том, что в основе зрительного восприятия лежат процессы сбора, представления, обработки и рас-

познавания информации, отражающей свойства наблюдаемого человеком реального мира. Зрение – это процесс определения по изображениям, что именно присутствует в окружающем мире и где именно оно находится, т. е. зрение сводится к решению задач обработки информации. Обработка изображений должна опираться на несколько последовательных уровней: от представления объектов (растровое изображение, неструктурированная информация) к их символическому представлению (данные в структурированной форме). Исходя из этого, в области машинного зрения принято выделять следующие основные этапы обработки данных:

- предобработка изображений;
- сегментация;
- выделение геометрической структуры;
- определение относительной структуры и семантики.

Мы будем применять модульную парадигму Д. Марра в обработке изображений.

Традиционный термин «обработка изображений» чаще употребляется в последние годы для обозначения обработки нижнего уровня, когда результатом обработки изображения снова является изображение. В то же время термин «понимание изображений» употребляется для обработки верхнего уровня, часто в контексте применения методов искусственного интеллекта. Методы обработки высокого уровня, относящиеся собственно к «пониманию изображений», по-прежнему представляют собой «вызов» для сообщества исследователей в области компьютерного зрения и искусственного интеллекта. Безусловно, перспектива создания будущих поколений «интеллектуальных машин» в основном зависит от дальнейшей разработки именно этого круга алгоритмов.

Сегодня теории обработки изображений во многом превратились в технологии, мы гораздо более скромно говорим о технической дисциплине под названием «машинное зрение». Это не означает, что в области обработки и анализа изображений не осталось открытых проблем, их огромное количество. Но признаком несомненной зрелости прикладной науки является то, что теперь эти вопросы всегда ставятся в практической плоскости, с учетом обязательных и близких перспектив технического внедрения.

## D. Предисловие

В данной книге описан один из понятных и самых простых путей обработки изображений. Этот путь может освоить любой из школьников. Рассматривается подробно процесс захвата и обработки изображений в соответствии с модульной парадигмой Д. Марра. Учитель сможет найти в книге уроки с подробным описанием обработки изображений в инженерной графической среде программирования LabVIEW. Подробного описания ос-

нов программирования в LabVIEW в этой книге нет. Тем, кто незнаком с программированием в этой среде, рекомендую прочитать книгу «Узнайте, как программировать на LabVIEW». В ней уроки для 5–6-го классов, все предельно лаконично и понятно.

В этой новой книге, как обычно в наших работах, подробно приведены пути поиска пиктограмм функций и схемы итоговых программ. У вас все получится! Не сомневайтесь. В разделе «Наши проекты» приведены темы и краткие описания проектов, которые выполнены нами в школе с ребятами 5–11-х классов..

На сайте издательства размещена демоверсия библиотеки средств обработки и анализа изображений IMAQ Vision блока **Vision and Motion** в среде LabVIEW.

Особую благодарность выражаю Михееву Павлу Михайловичу.

Михеев П. М., кандидат физико-математических наук, был доцентом физического факультета МГУ имени М. В. Ломоносова, организатором и руководителем одного из старейших и лучших академических центров National Instruments в России, в настоящее время – «Центр измерительных технологий и промышленной автоматизации» на базе Лазерного центра МГУ. Именно он открыл для меня около 10 лет назад интересный, сложный и такой, с помощью LabVIEW, простой в работе мир цифровой обработки изображений [4].

А теперь попробуйте и вы. И у вас все обязательно получится.

# Регистрация цветного изображения с USB-камеры

УРОК 2

*В этом уроке рассказывается о том, как осуществить захват цветного изображения с USB-камеры. Рассмотрим необходимые этапы программирования захвата изображения. Познакомимся со способами отображения захваченного изображения на лицевой панели. Научимся записывать и читать графические файлы и оценим время подготовки к захвату изображения и длительность регистрации изображения.*

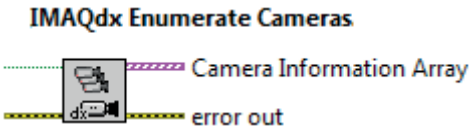
*В этом уроке изложены вопросы:*
*E. Доступные для программирования камеры*
*F. Захват изображения*
*G. Отображение изображения на лицевой панели*
*H. Запись и чтение изображения из файла*
*I. Оцениваем время захвата изображения*

## A. Доступные для программирования камеры

Для программирования USB-камеры надо указать имя камеры. Перечень допустимых камер можно получить с помощью функции IMAQ USB Enumerate Cameras.

Выходной параметр этой функции – строковый массив с именами и характеристиками доступных камер.

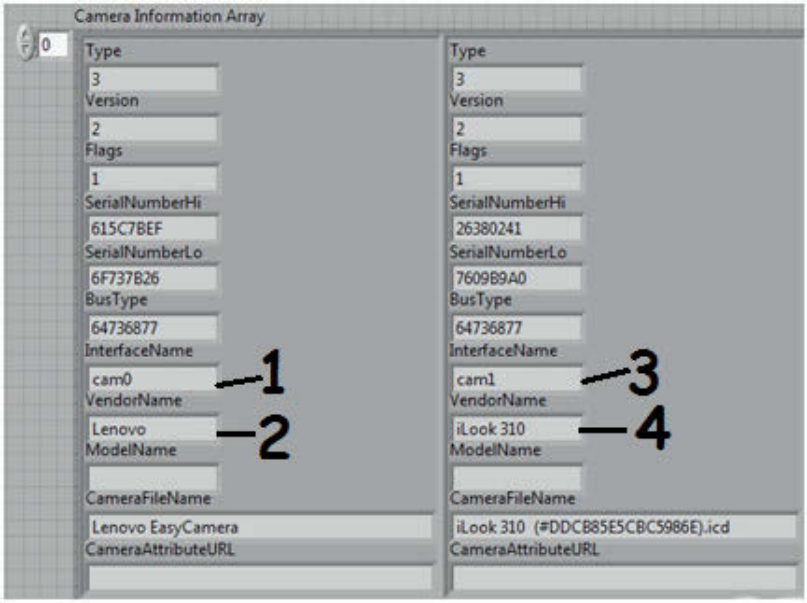
Путь к функции выдачи списка доступных камер: **Vision and Motion** ⇨ **NI-IMAQ dx** ⇨ **IMAQ Enumerate Cameras**.




Наводим мышку на правую верхнюю контактную площадку **Camera Information Array** (она розового цвета) и после щелчка правой кнопкой выбираем в контекстном меню **Create** ⇨ **Indicator**.

Получим программу lesson2_1.vi:


На лицевой панели возникнет индикатор для отображения информации. Это массив строковых кластеров с характеристиками доступных камер (см. рисунок ниже). После запуска программы виртуального прибора можно анализировать информацию. Для программирования, чтобы снимать изображения с нужной камеры, надо точно выбрать имя этой камеры из предложенного списка камер в той или иной используемой функции обработки изображений. В примере в массиве строковых кластеров отображены две камеры, подключенные в данный момент к компьютеру.




1 – cam0, имя камеры для программирования; 2 – Lenovo, встроенная камера; 3 – cam1, имя камеры для программирования; 4 – iLook 310, модель подключенной веб-камеры

## B. Захват изображения

Процесс захвата изображения – сохранение характеристик изображения в памяти компьютера. Пока не будем рассматривать, какие характеристики изображения мы сохраняем в памяти PC. Об этом поговорим позже.

Процесс захвата состоит из нескольких этапов:

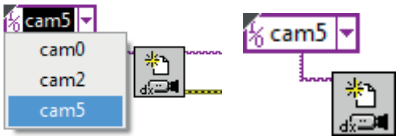
1) инициализация камеры;
2) подготовка буфера для изображения;
3) запуск сессии захвата;
4) непосредственно захват изображения;
5) закрытие сессии захвата изображения.

Рассмотрим эти этапы подробнее.

### *1. Инициализация камеры*

Путь к функции инициализации камеры: **Vision and Motion** ⇨ **NI-IMAQ dx** ⇨ **Open**.

Из списка доступных видеокамер выберем одну, например с именем **cam5**. Для этого на левой верхней контактной площадке (она сиреневого цвета) щелкнем правой кнопкой мыши и из контекстного меню выберем **Create** ⇨ **Constant**. Далее из списка констант выберем имя **cam5**.



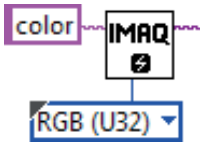

При инициализации камеры происходит открытие ее для работы, создается уникальная ссылка для этой камеры для использования этой ссылки при программировании.

Эту функцию в программе надо выполнять один раз для данной камеры.

### *2. Подготовка буфера для изображения*

Буфер изображения – ссылка на область памяти, где планируется хранить изображение. Для резервирования определенным образом устроенной области памяти надо указать прежде всего имя этой области (параметр **Image Name**) и принцип организации хранения изображения, – т. е. формат изображения (параметр **Image Type**).

**Vision and Motion** ⇨ **Vision Utilities** ⇨ **Image Management** ⇨ **IMAQ Create**.




Аналогично задаем параметры **Image Name** и **Image Type** функции подготовки буфера изображения **IMAQ Create**. Чтобы осуществить эту опе-

рацию, надо щелкнуть правой кнопкой мыши по контактной площадке параметра **Image Type**, в контекстном окне выбрать **Create ⇨ Constant** и нужный тип изображения: в нашем примере **Image Type = RGB (U32)**. Чтобы задать имя буфера изображения, надо щелкнуть правой кнопкой мыши по контактной площадке параметра **Image Name**, в контекстном окне выбрать **Create ⇨ Indicator** и в образовавшемся окошке ввести имя. В нашем примере **Image Name = color**.

Чтобы иметь возможность работать с несколькими изображениями, полученными с одной камеры, надо для каждого из них задать буфер изображения с уникальным именем и соответствующим типом. Эта операция может быть выполнена в программе многократно нужное число раз.

### *3. Запускается сессия захвата*

Этот процесс организовывает, подготавливает саму сессию захвата изображения.

**Vision and Motion ⇨ NI-IMAQ dx ⇨ Configure Grab**.

Соединим выходной параметр **Session Out** сессии захвата функции **Open** с входным параметром **Session In** функции **Configure Grab**. Тем самым мы упорядочиваем процесс подготовки к захвату изображения. Наладим последовательные соединения входных и выходных параметров для кластера ошибок.

Эту функцию в программе надо выполнять один раз для данной камеры.

### *4. Непосредственно захват изображения*

**Vision and Motion ⇨ NI-IMAQ dx ⇨ Grab**.

Соединим входной параметр исходного изображения **Image In** для функции **Grab** с выходным параметром **New Image** функции инициализации буфера изображения **IMAQ Create**. Этим мы определим, какого типа мы будем формировать изображение с камеры и где оно будет предварительно сохранено.

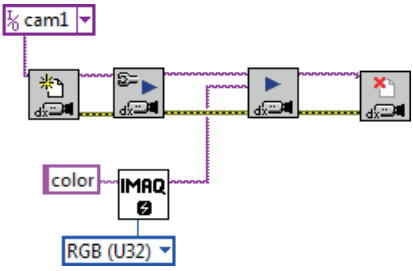
Эта операция может быть выполнена в программе многократно нужное число раз.

### *5. Закрываем сессию захвата изображения*

**Vision and Motion ⇨ NI-IMAQ dx ⇨ Close**.

Эту функцию в программе надо выполнять один раз для данной камеры. Она делает недоступной работу с камерой с конкретным именем **cam5**.

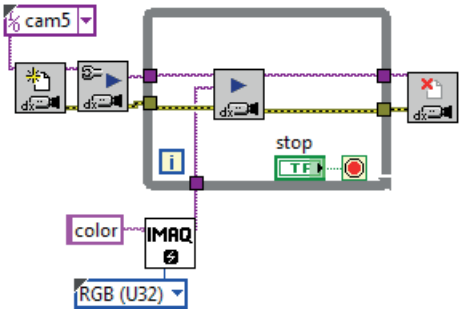
В итоге получили следующую программу lesson2_2.vi.




После запуска программы захваченное изображение будет храниться в буфере с именем **color**, и его можно использовать в дальнейшем для обработки, но пока мы не можем увидеть изображение, отобразить его на лицевой панели.

Перепишем программу для многократного отображения изображения с веб-камеры. Для этих целей используем структуру цикла с выходом по условию. В качестве окончания выполнения цикла будем отслеживать нажатие кнопки **Stop**. Понятно, что в цикле необходимо выполнять только непосредственно захват изображения.

Финальная программа **lesson2_2.vi** имеет вид:

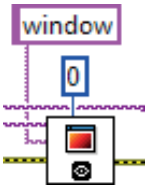


# C. Отображение изображения на лицевой панели

Отображение изображения на экране компьютера можно выполнить двумя способами.

1. Создать отдельное окно для отображения изображения с помощью функции **IMAQ WindDraw**. Можно создать до 16 различных окон.

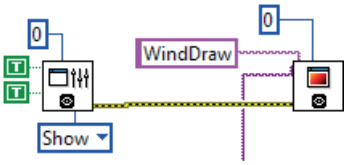
**Vision and Motion ⇨ Vision Utilities ⇨ External Display ⇨ IMAQ WindDraw (0** – номер окна, **window** – название окна).



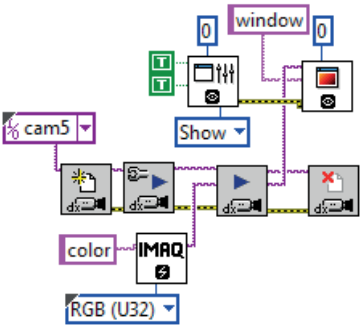

Предварительно можно задать параметры этого окна с помощью функции **IMAQ WindSetup**. Путь к этой функции: **Vision and Motion ⇨ Vision Utilities ⇨ External Display ⇨ IMAQ WindSetup**.

В частности, удобно организовать окно с прокруткой по горизонтали и вертикали.

Получим фрагмент программы:




Блок-диаграмма всей программы **lesson2_3.vi** имеет вид:

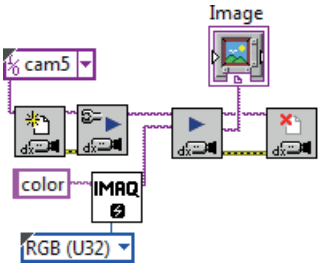


2. Создать окно для отображения изображения можно непосредственно на лицевой панели ВП.

Перейдем в окно лицевой панели, щелкнем правой кнопкой мыши по полю лицевой панели и выберем в окне элемент отображения: **Controls** ⇨ **Vision** ⇨ **Image Display**, индикатор окна отображения изображения.


На блок-диаграмме при этом появится пиктограмма этого индикатора. Осталось лишь подсоединить в окне блок-диаграммы индикатор **Image Display** с выходным параметром **Image Out** функции **Grab**.

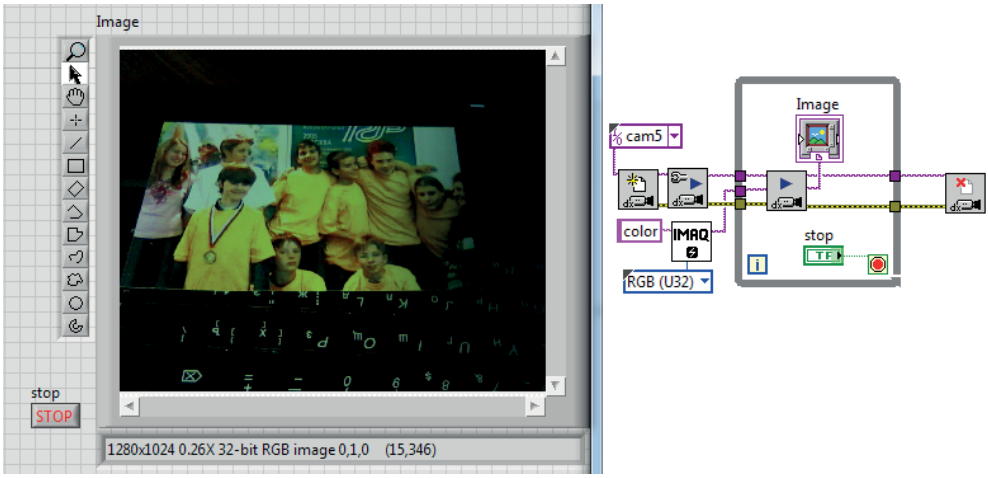
Программа захвата цветного изображения с USB-камеры с отображением его на экран в окне на лицевой панели выглядит следующим образом:




Получим наиболее часто используемую программу **lesson2_4.vi**.

Если щелкнуть правой кнопкой мыши по **Image Display** на лицевой панели и выбрать **Zoom to Hit**, то можно масштабировать захваченное изображение, чтобы его размер совпал с размером **Image Display** на лицевой панели. В этом случае все изображение полностью отобразится в окне вывода **Image Display**.

Осталось создать цикл для многократного захвата и отображения изображения на лицевой панели. Напомним, что внутри циклической структуры надо разместить только функцию захвата изображения и индикатор, отображающий захваченное изображение, на лицевой панели.



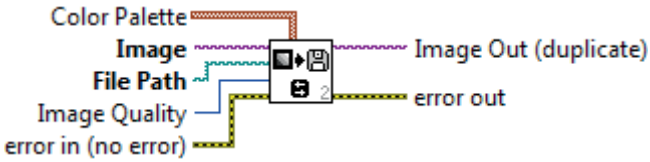

## D. Запись и чтение изображения из файла

Захваченное изображение можно записать в файл в нужном формате: BMP, JPG, PNG, TIFF. Для записи изображения в файл надо в ВП lesson1_4.vi добавить функции открыть файл, записать изображение в файл и закрыть файл.

В палитре функций **Programming** ⇨ **File I/O** взять функции **Open/Create File** и **Close File**. Для функции **Open/Create File** создадим терминал для ввода пути уже существующего файла. Ввести в него абсолютный путь к файлу, в котором будем хранить изображение.

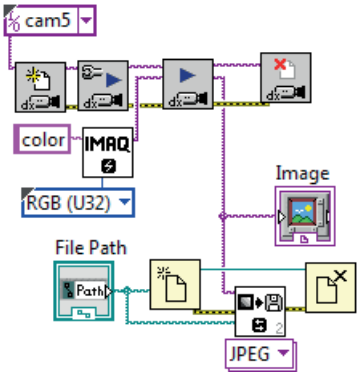
В палитре функций **Vision and Motion** ⇨ **Vision Utilities** ⇨ **Files** возьмем функцию **IMAQ Write File**:

Контактную площадку **Image** соединим с захваченным с помощью функции **Grab** изображением **Image**.




Параметр **File Path** возьмем с терминала при открытии файла.

Подсоединим кластер ошибок. Добавим этот фрагмент программы в ВП lesson2_4.vi и получим программу записи захваченного изображения в файл, например в формате JPEG.




Сохраним блок-диаграмму этой программы под именем lesson2_5.vi.

Чтобы прочитать изображение из файла, надо открыть файл с изображением, в палитре функций **Vision and Motion** ⇨ **Vision Untities** ⇨ **Files** взять функцию **IMAQ Read File** и затем закрыть файл. Отметим, что для отображения изображения из файла на лицевой панели надо установить элемент отображения: **Controls** ⇨ **Vision** ⇨ **Image Display**, индикатор окна отображения изображения.


Создайте эту несложную программу самостоятельно.

## E. Оцениваем время захвата изображения

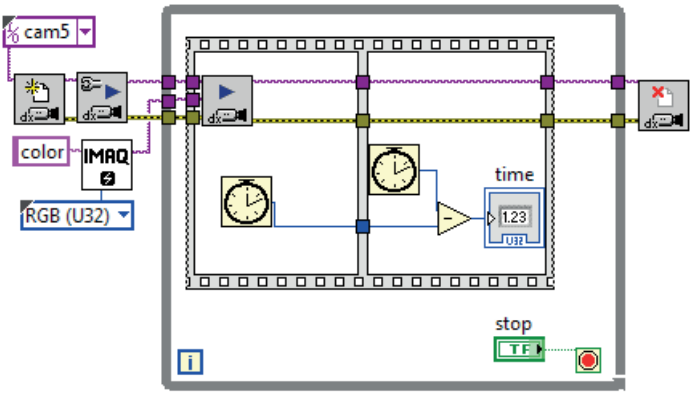
Для осуществления управления роботом операцию захвата изображения необходимо выполнять многократно. Оценим время выполнения функции регистрации изображения.

Рассмотрим вариант программы lesson1_4.vi с использованием цикла. Для оценки интересующего нас времени нужно вынести из цикла отображение изображения, т. е. убрать индикатор окна отображения изображения.


Теперь наша задача – найти время выполнения одной итерации цикла. Приведем два варианта программы. В одной программе предлагается использовать последовательную структуру фреймов, в другой – использовать сдвиговые регистры.

С использованием последовательной структуры мы сможем отследить время начала захвата изображения и время окончания регистрации. Напомним, что потоковый принцип обработки данных предполагает одновременно начинать регистрацию изображения и фиксировать время с системного таймера в миллисекундах. После того как процесс захвата изображения завершился, начинает выполняться второй фрейм последовательной структуры, где фиксируется вновь время таймера. Разность

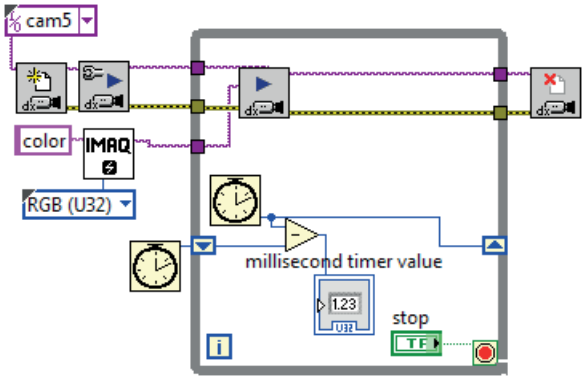
значений времени выводится на индикаторе в цикле. В нашем случае эта величина составила 0,3 сек.




Сохраним блок-диаграмму этой программы под именем lesson2_6.vi.

Теперь рассмотрим вариант программы с использованием сдвиговых регистров.

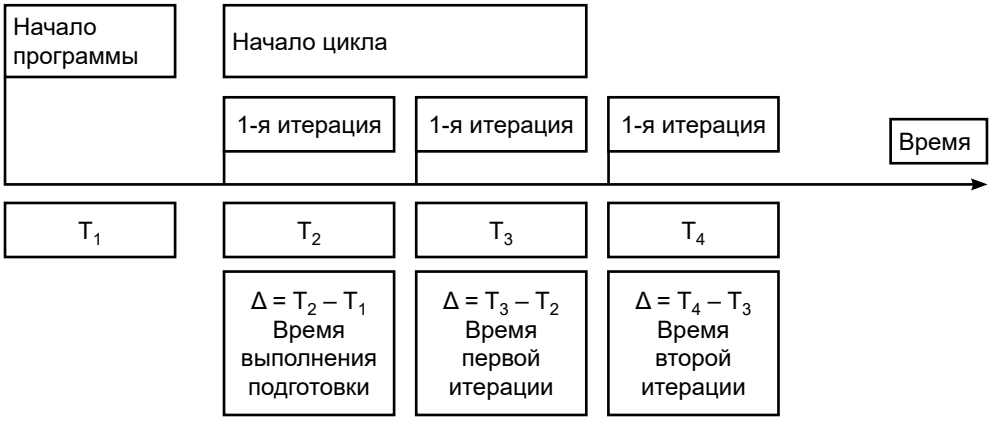
Создадим на границах цикла сдвиговый регистр.




Отметим, что за первую итерацию цикла мы не получим точного результата. В цикле таймер фиксирует время начала итерации. После того как выполнятся функции инициализации камеры, подготовки буфера для изображения, подготовки сессии захвата, начинается цикл. Первое показание индикатора времени можно считать временем подготовки к регистрации изображения. Оно порядка 2,5 сек, это в почти в 10 раз больше, чем время регистрации изображения. Во всех последующих итерациях таймер в цикле фиксирует время начала итерации и передает его на следующую итера-

цию. Таким образом, на каждой итерации можно вычислить длительность предыдущей итерации.

Сохраним блок-диаграмму этой программы под именем lesson2_7.vi.



# Обработка цветных изображений

УРОК 3

*В этом уроке рассказывается о том, как осуществить оцифровку цветного изображения с USB-камеры. Рассматриваются программы получения цифровых 2D-матриц для всех цветовых компонент пикселя изображения. Познакомимся с программой распознавания цвета пикселя сцены, выбранного манипулятором мышь. Рассмотрим две цветовые модели RGB и HSL.*

*В этом уроке изложены вопросы:*

*J. Цветные изображения*
*K. Распознавание цвета пикселя RGB-изображения*
*L. Программа автоматического распознавания цвета пикселя RGB-изображения*
*M. Распознавание цвета пикселя HSL-изображения*

## A. Цветные изображения. Цветовая модель RGB

Вы, наверное, заметили, какими разными могут быть изображения! Прежде всего цветные изображения. Очень похожие на цветные, только без разноцветных красок. Такие изображения называют изображениями в градациях серого. В них присутствуют различные оттенки серого цвета от очень темного серого – черного до совершенно светлого серого – белого. И наконец, встречаются черно-белые изображения, состоящие только из белого и черного цветов, т. е. из двух цветов. Такие изображения называют бинарными. Цвета могут быть не только черный и белый, но и другие любые, например красный и черный, но их в изображении обязательно только два.

Прежде чем начать работать с изображениями разных типов, надо понять, как они хранятся в памяти PC, т. е. познакомиться с их форматом. Напомним, что вся память PC разбита на элементарные ячейки, которые называются битами. С помощью одного бита можно запомнить два разных значения – 0 или 1. Биты группируются по восемь, в октеты, и такие группы называют байтами. Все байты памяти PC пронумерованы, т. е. имеют свой адрес. Но что для нас важнее, так это то, что с помощью одного байта можно запомнить 256 разных значений, от 0 до 255 ($256 = 2^8$).

Большинство изображений с физической точки зрения представляет собой зарегистрированное специальным датчиком двумерное распределение интенсивности электромагнитного излучения, отраженного объектом реги-