



Video Tracking System Based on DirectShow

Deng Fuqin

He Jinshou

Zhang Chong

Contents

- Objective
- Why Use DirectShow?
- How to Use DirectShow For Video Tracking?
- Video Tracking Algorithms
- Experiment Results & Discuss

Objective

- An Object is moving in a video, we want to track it by computer
 - the video can be a video file, such as avi, wmv, rm, etc. or can be from a live video input such as a camera.
 - The right side is some images from videos. Our Objective is to track the light or the hand real time.



Why Use DirectShow?

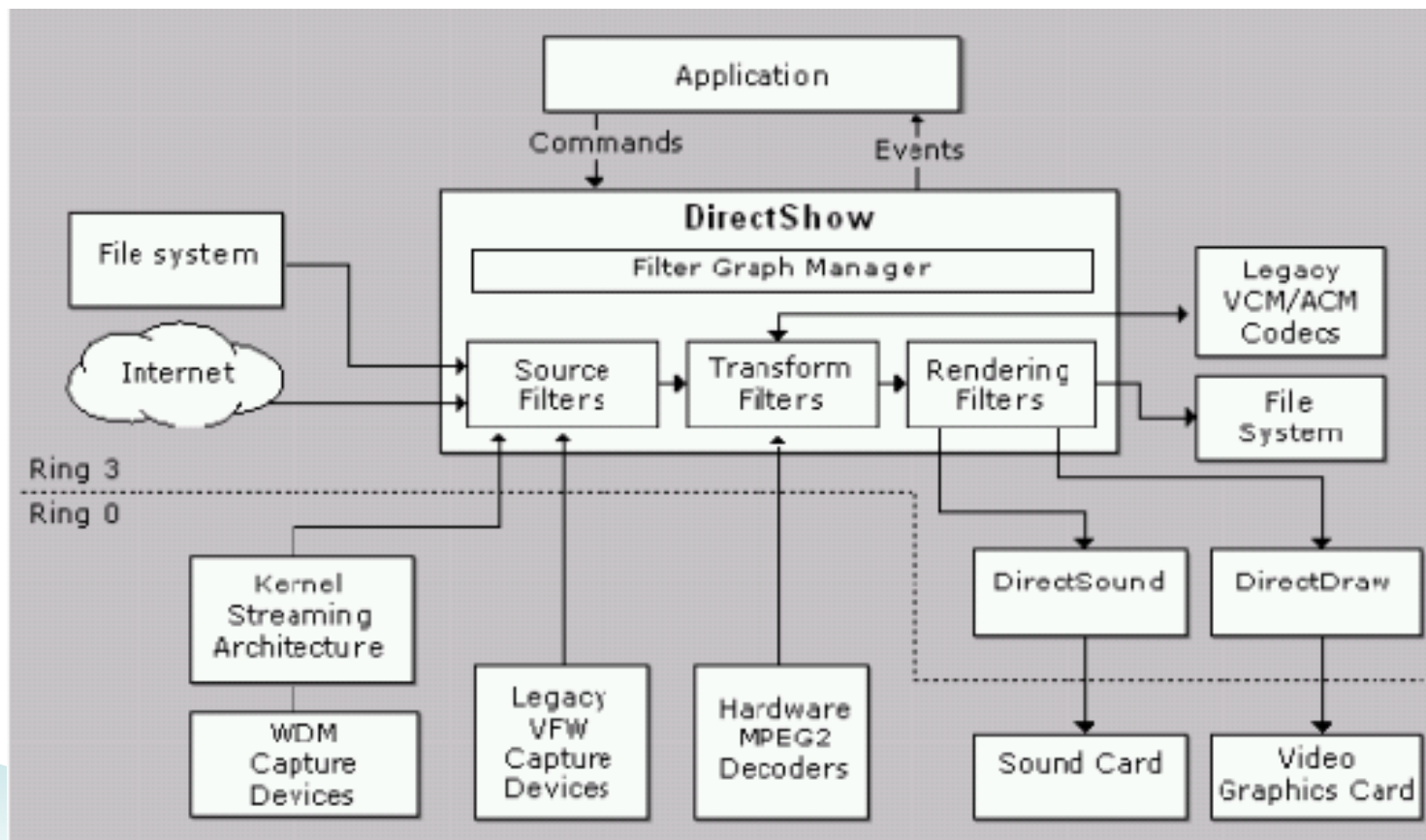
Working with multimedia presents several major challenges:

- Multimedia streams contain large amounts of data, which must be processed very quickly.
- Audio and video must be synchronized so that it starts and stops at the same time, and plays at the same rate.
- Data can come from many sources, including local files, computer networks, television broadcasts, and video cameras.
- Data comes in a variety of formats, such as Audio-Video Interleaved (AVI), Advanced Streaming Format (ASF), Motion Picture Experts Group (MPEG), and Digital Video (DV).
- The programmer does not know in advance what hardware devices will be present on the end-user's system.

Why Use DirectShow?(cont)

- DirectShow is designed to address each of these challenges.
 - Its main design goal is to simplify the task of creating digital media applications on the Windows® platform, by isolating applications from the complexities of data transports, hardware differences, and synchronization.
 - To achieve the throughput needed to stream video and audio, DirectShow uses DirectDraw® and DirectSound® whenever possible. These technologies render data efficiently to the user's sound and graphics cards.
 - DirectShow synchronizes playback by encapsulating media data in time-stamped samples.
 - To handle the variety of sources, formats, and hardware devices that are possible, DirectShow uses a modular architecture, in which the application mixes and matches different software components called **filters**.

- the relationship between an application, the DirectShow components, and some of the hardware and software components that DirectShow supports



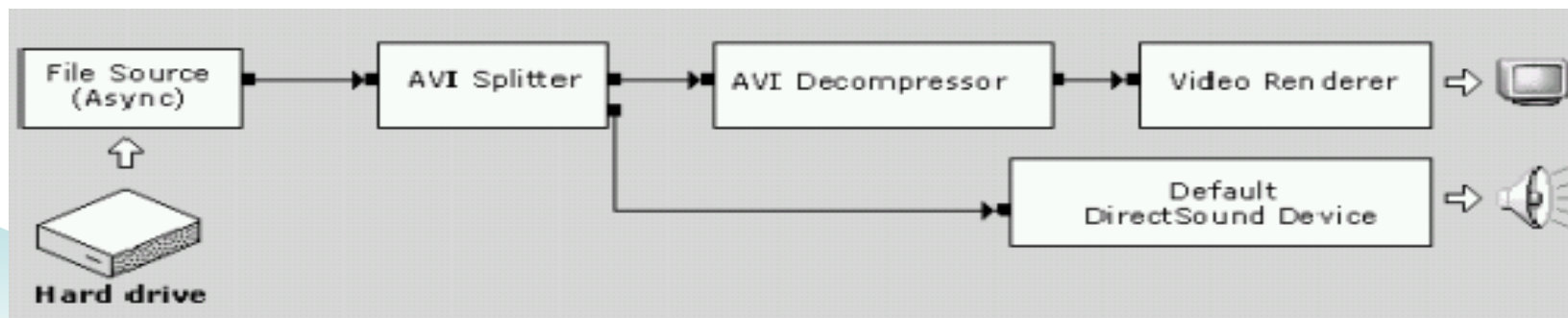
How to Use DirectShow For Video Tracking?

– Preparation

- Install Visual C++ 6.0
- Install DirectX SDK 9.0, you can download it from Microsoft's website
- Setup the Include Path and Lib Path of VC
 - In VC, Select menu tools → options, then select option group “directories”, Select “Include files”, Add the following path :” X:\DXSDK\Include” and “X:\DXSDK\Samples\C++\DirectShow\BaseClasses”. Select “library files”, and add the path below:” X:\DXSDK\Lib”
 - Open “X:\DXSDK\Samples\C++\DirectShow\BaseClasses\baseclasses.dsw”, compile it ,and copy the files “STRMBASE.lib” and “STRMBASD.lib” to F:\DXSDK\Lib
- Now, we setup our development environment successfully.

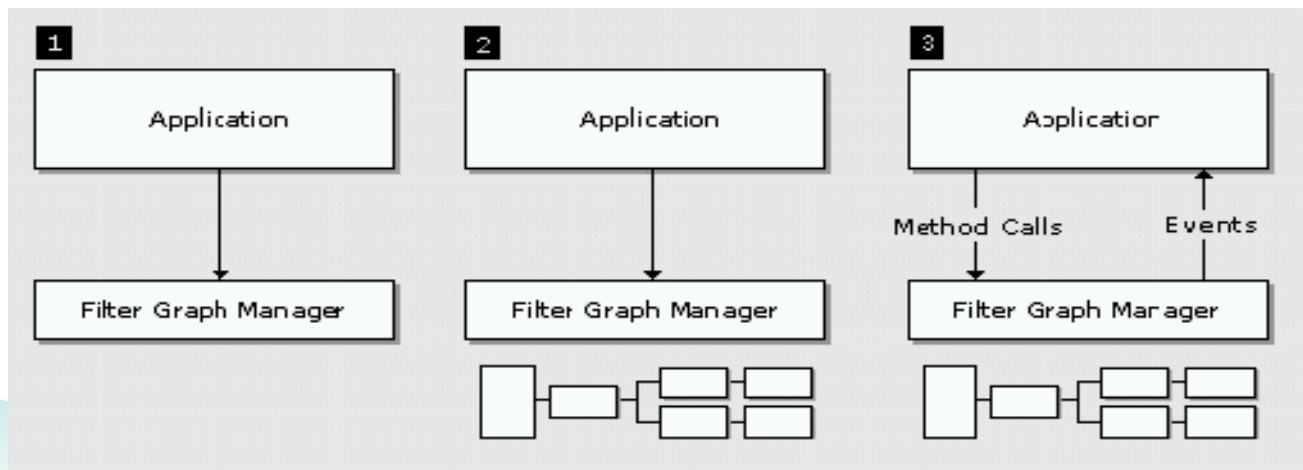
Filters and Filter Graphs

- The building block of DirectShow is a software component called a filter. A filter is a software component that performs some operation on a multimedia stream.
- For example, DirectShow filters can
 - read files
 - get video from a video capture device
 - decode various stream formats, such as MPEG-1,2,4 video ,RM video
 - pass data to the graphics or sound card
- Filters receive input and produce output. For example, if a filter decodes MPEG-1 video, the input is the MPEG-encoded stream and the output is a series of uncompressed video frames.
- In DirectShow, an application performs any task by connecting chains of filters together, so that the output from one filter becomes the input for another. A set of connected filters is called a **filter graph**. For example, the following diagram shows a filter graph for playing an AVI file.



– Writing a DirectShow Application

- The application creates an instance of the **Filter Graph Manager**.
- The application uses the Filter Graph Manager to build a filter graph. The exact set of filters in the graph will depend on the application.
- The application uses the Filter Graph Manager to control the filter graph and stream data through the filters. Throughout this process, the application will also respond to events from the Filter Graph Manager.
- When processing is completed, the application releases the Filter Graph Manager and all of the filters.



– three kinds of filters:

– Source filter

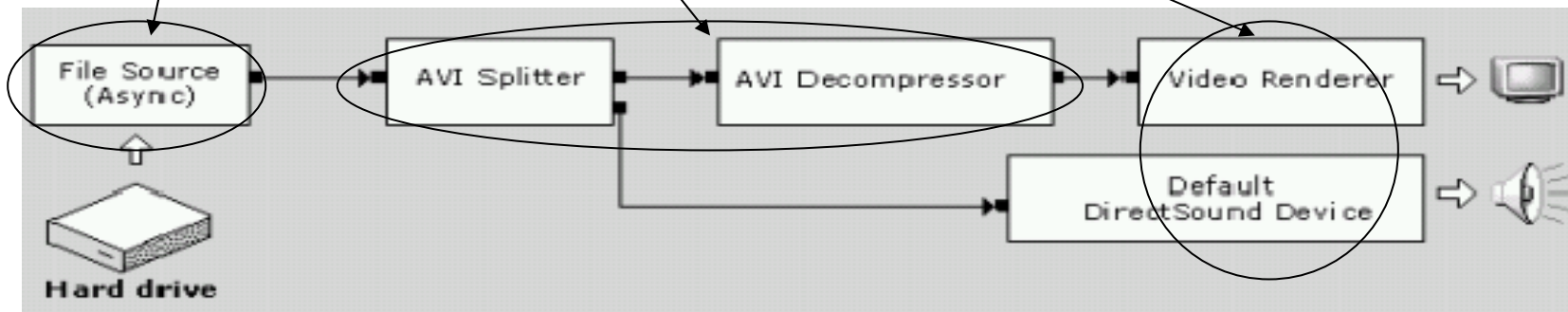
- Get multimedia data from files, Internet, video capture cards, or cameras


– Transform filter

- Encode and Decode of multimedia data,

– Renderer filter

- Send the data to a video card, or internet, or save to a file



- 
- How to write a filter?
 - Step 1. Choose a Base Class
 - Step 2. Declare the Filter Class
 - Step 3. Support Media Type Negotiation
 - Step 4. Set Allocator Properties
 - Step 5. Transform the Image
 - Step 6. Add Support for COM
 - More Information, see MSDN...

Video Tracking Algorithms

- Color Space
- Tracking Algorithms Based on Brightness
 - Rectangle algorithm
 - An Improved Algorithm
- Tracking Algorithms Based on Static Background

Color Space

- The feature that we will be looking at for this demo application is brightness, and we will track objects based on their brightness.
- YUV is one color space that has this very component that we are seeking for
- a conversion is required from the typical RGB24 input format to YUV
 - $Y = 0.299R + 0.587G + 0.114B$
 - $U = -0.148R - 0.289G + 0.437B$
 - $V = 0.615R - 0.515G - 0.100B$
- For more fast computing speed, we use $Y = (R+G) >> 1$ to compute the brightness.

Rectangle algorithm

- The rectangle algorithm keeps track of four points in each frame, the top most, left most, right most and bottom most points where the brightness exceeds a certain threshold value.

```
Initialize left=inf,right=0,top=inf,bottom=0
For each pixel p(x,y)
    Compute the brightness of the pixel use  $Y=(R+G)>>1$ 
    If the brightness is larger enough ,then
        If left>x then left=x
        If right<x then right=x

        If top>y then top=x
        If bottom<y then bottom=y
    End if
End for
```

Rectangle algorithm

- A rectangle can be constructed from these points, which tells us where the bright object is. The border of the rectangle is then simply replaced by a predefined color.

For x=left to right

Mark pixel $p(x, \text{top})$ with RED color.

Mark pixel $p(x, \text{bottom})$ with RED color.

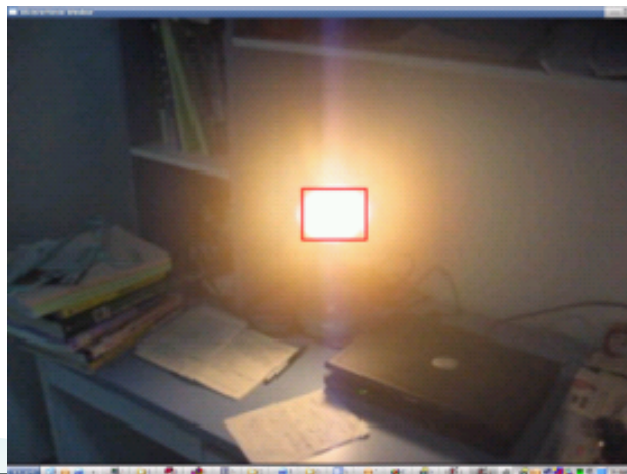
End if

For y=top to bottom

Mark pixel $p(\text{left}, y)$ with RED color.

Mark pixel $p(\text{right}, y)$ with RED color.

End if



An Improved Algorithm

- This algorithm tracks objects by identifying segments that make up the object on the screen. Each segment consists of the head and the length of the segment. The object is constructed by grouping the segments together.

```
Init an empty list which consists the struct QSEG{x,y,length} .
For y=1 to height of the image
    QSEG Qseg={0,0,0};
    For x=1 to width of the image
        If the brightness of p(x,y) > threshold ,then
            If sqseg.length==0 then
                Qseg.x=x
                Qseg.y=y
                Qseg.length=1
            Else
                Qseg.length++
            End if
        End if
    End for
    Add Qseg to the list.
End for
```


An Improved Algorithm

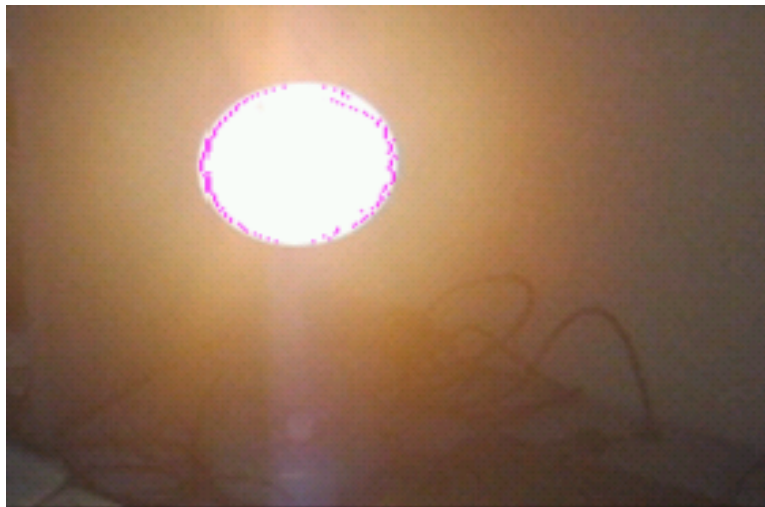
- Now we can draw the object

For each element qseg in the list

Mark pixel $p(qseg.x, qseg.y)$ with RED color

Mark pixel $p(qseg.x+length, qseg.y)$ with RED color

End for



Tracking Algorithms Based on Static Background

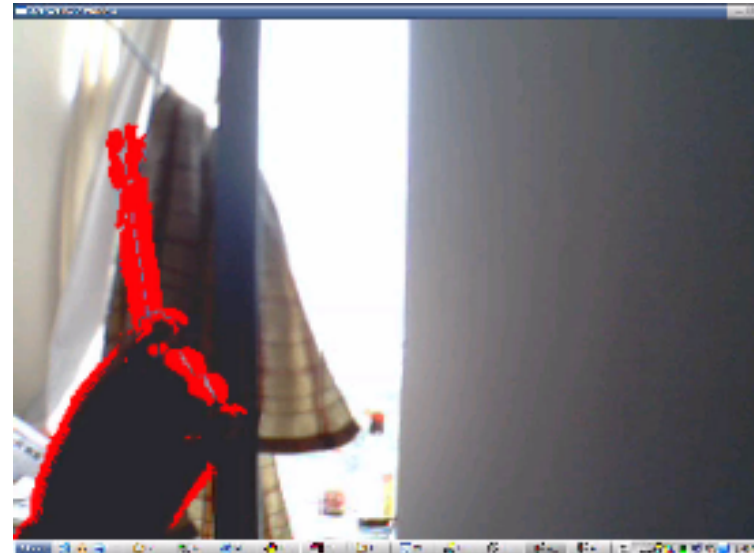
- If the background is not moving, then we can compute the brightness differences between the foreground and the background. If the difference is Large Enough, then we treat it as something stands there.
- Main Frames of tracking algorithm:

```
For each pixel in one frame image
    Compute the difference of pixel values between
    background image and foreground image
    If the difference is larger than the threshold value, then
        Make a mark on the pixel which denotes the pixel
        as object tracked
    End if
End for
```

Tracking Algorithms Based on Static Background

- There are some kinds of ways to get the background image:
 - compare the current frame with the previous one. It's useful in video compression when you need to estimate changes and write only the changes, not the whole frame. Also it is an effective and efficient method for video tracking.
 - Things become worse, when the object moves slowly
 - The background image is given before tracking,
 - this is the easiest way. But it often works not as good as we think, since the image in the video is always changing for the changing of weather or the camera's parameters
 - The background image is calculated from the video by algorithm.
 - There are many different algorithms for computing the background images, but most of them are time consuming. We use a relative simple algorithms to compute the background image, results shows the algorithms is effect, but still needs improving.

Tracking Algorithms Based on Static Background



Experiment Results & Discuss

- The two kinds of algorithms are suit for different application fields.
 - The first algorithm is based on brightness,It can be applied in tracking very white or very black objects whose color are very distinct from the background.
 - The second algorithm is based on stable background and can only tracks moving objects.It can be used in such fields where the camera is not moving,such as traffic monitoring,hotel monitoring,and so on.
- There are still many problems need improving,such as:
 - The algorithms just use pixel gray value,do not use other information,such as object geometry structure,which is always used by our human's vision system.
 - In the second algorithm,if the background can not obtained directly,we must extract it from the video,This is a very hard work,The methods in this paper is VERY preliminary.The algorithms we introduce are still need improving.



Thank you very much!