

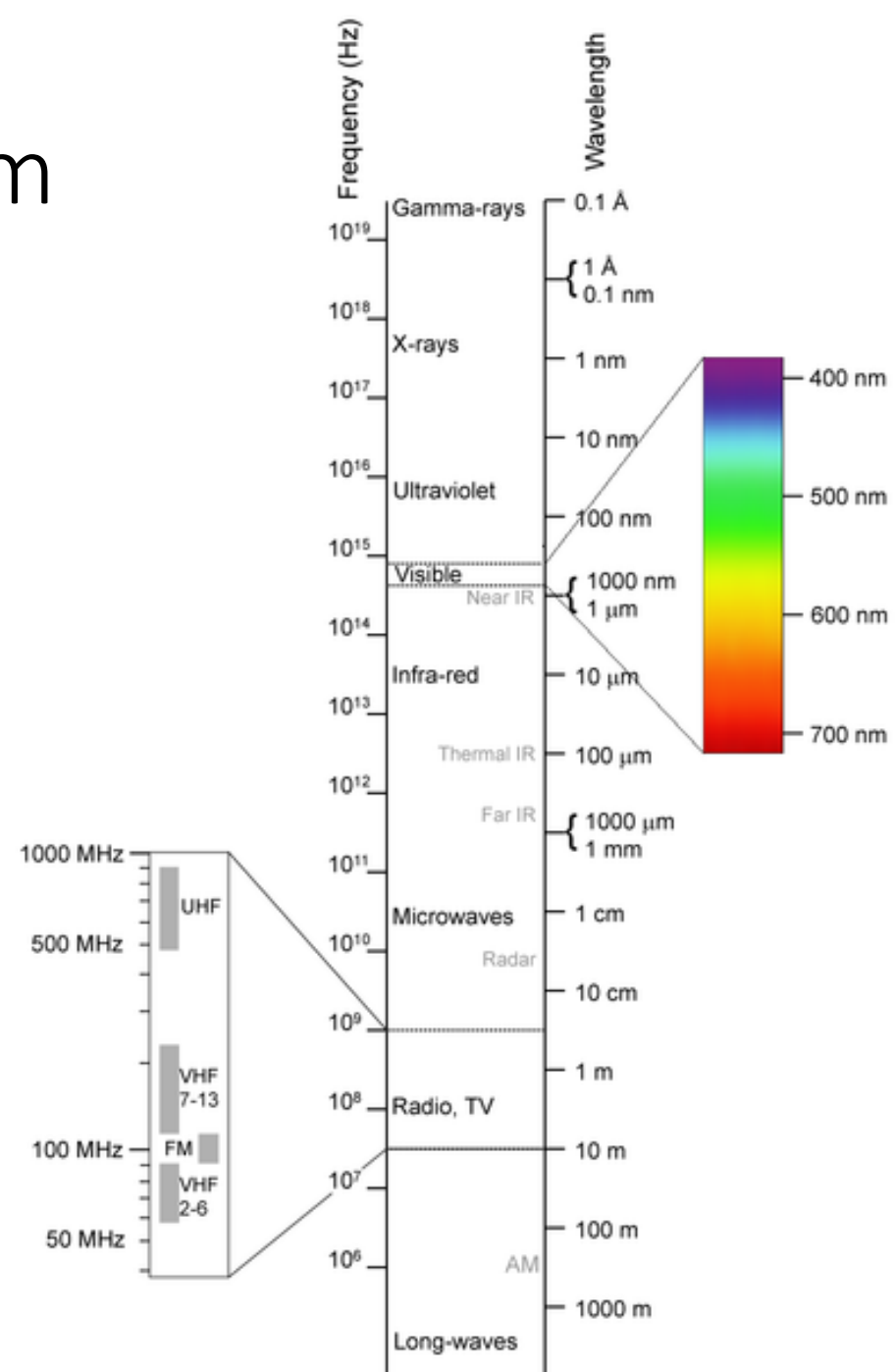
Video Compression

CS6025 Data Encoding

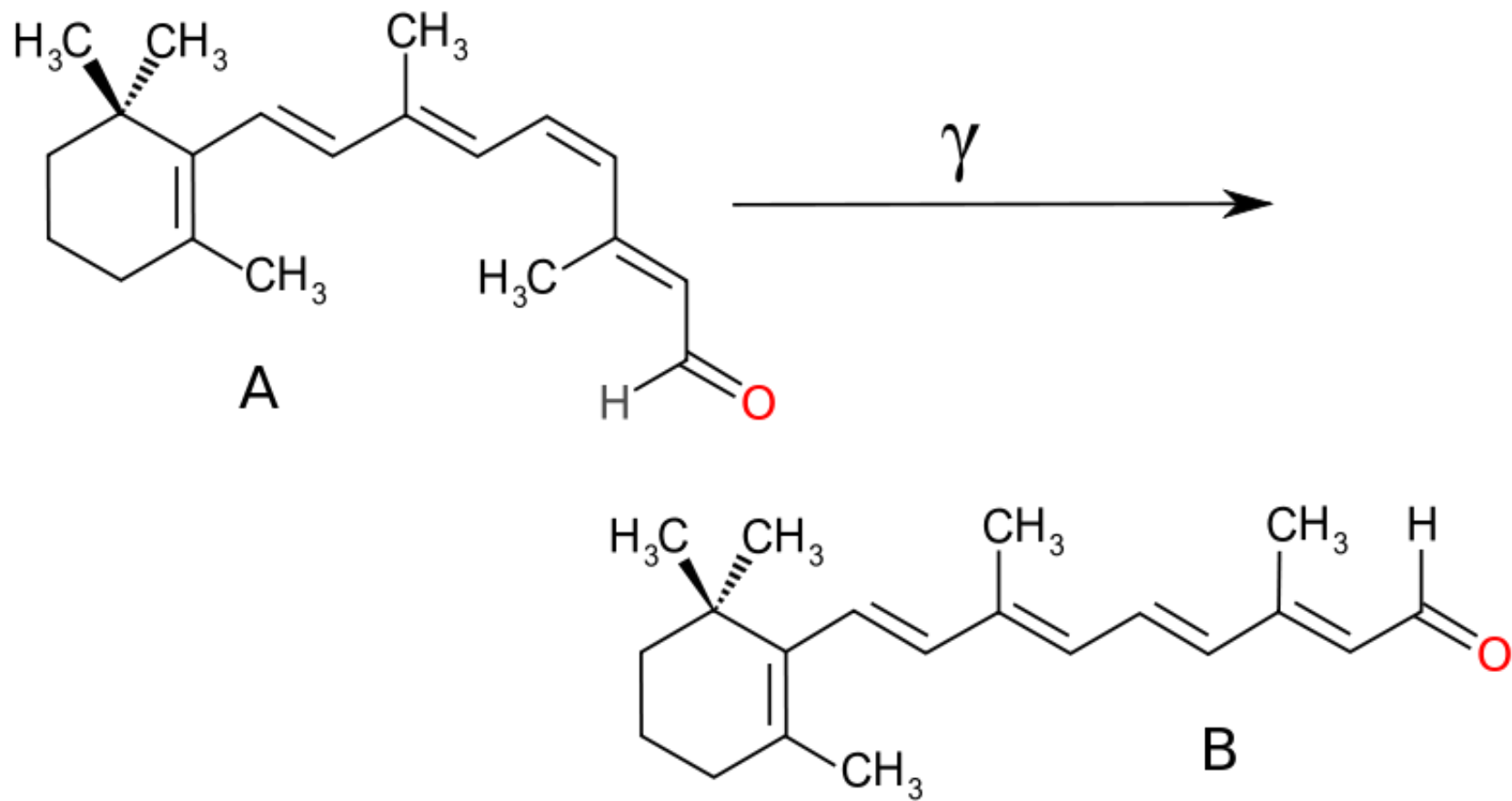
Yizong Cheng

4-21-15

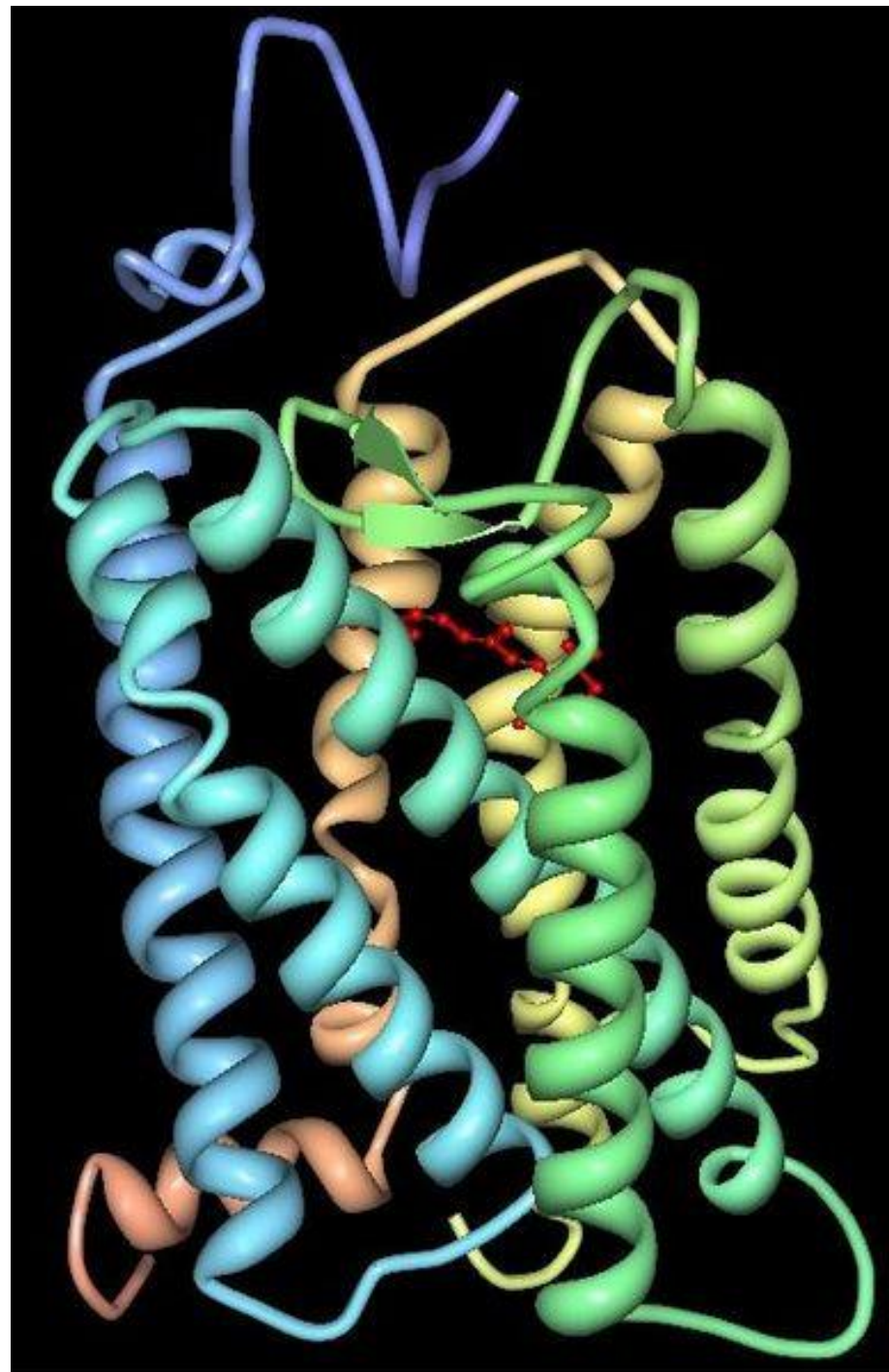
Frequency Spectrum



Retinal



Opsin



Gene RHO on Chromosome 3

RHO rhodopsin [Homo sapiens]

Gene ID: 6010, updated on 7-Feb-2013

Summary

Official Symbol RHO provided by HGNC

Official Full Name rhodopsin provided by HGNC

Primary source HGNC:10012

See related Ensembl:ENSG00000163914; HPRD:01584; MIM:180380; Vega:OTTHUMG00000159542

Gene type protein coding

RefSeq status REVIEWED

Organism Homo sapiens

Lineage Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo

Also known as RP4; OPN2; CSNBAD1

Summary Retinitis pigmentosa is an inherited progressive disease which is a major cause of blindness in western communities. It can be inherited as an autosomal dominant, autosomal recessive, or X-linked recessive disorder. In the autosomal dominant form, which comprises about 25% of total cases, approximately 30% of families have mutations in the gene encoding the rod photoreceptor-specific protein rhodopsin. This is the transmembrane protein which, when photoexcited, initiates the visual transduction cascade. Defects in this gene are also one of the causes of congenital stationary night blindness. [provided by RefSeq, Jul 2008]

Genomic context

Location: 3q21-q24

Sequence: Chromosome: 3; NC_000003.11 (129247482..129254187)

See RHO in [Epigenomics](#), [MapViewer](#)

Chromosome 3 - NC_000003.11

[129149793] [129325582]

MBD4 IFT122 RHO PLXND1 HLF00

Genomic regions, transcripts, and products

[Summary](#)

[Genomic context](#)

[Genomic regions, transcripts, and products](#)

[Bibliography](#)

[Phenotypes](#)

[Interactions](#)

[General gene info](#)

[General protein info](#)

[Reference sequences](#)

[Related sequences](#)

[Additional links](#)

Related information

[Order cDNA clone](#)

[BioAssay](#)

[BioProjects](#)

[BioSystems](#)

[CCDS](#)

[Conserved Domains](#)

[dbVar](#)

[Full text in PMC](#)

[Genome](#)

[GEO Profiles](#)

[GTR](#)

[HomoloGene](#)

[Map Viewer](#)

[MedGen](#)

[Nucleotide](#)

[OMIM](#)

OPN1LW on the X Chromosome

OPN1LW opsin 1 (cone pigments), long-wave-sensitive [*Homo sapiens*]
Gene ID: 5956, updated on 3-Feb-2013

Summary

Official Symbol OPN1LW provided by HGNC
Official Full Name opsin 1 (cone pigments), long-wave-sensitive provided by HGNC
Primary source [HGNC:9936](#)
Locus tag hCG_41347
See related [Ensembl:ENSG00000102076](#); [HPRD:02366](#); [MIM:300822](#); [Vega:OTTHUMG00000034295](#)
Gene type protein coding
RefSeq status REVIEWED
Organism [Homo sapiens](#)
Lineage Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo
Also known as CBP; RCP; ROP; CBBM; COD5
Summary This gene encodes for a light absorbing visual pigment of the opsin gene family. The encoded protein is called red cone photopigment or long-wavelength sensitive opsin. Opsins are G-protein coupled receptors with seven transmembrane domains, an N-terminal extracellular domain, and a C-terminal cytoplasmic domain. This gene and the medium-wavelength opsin gene are tandemly arrayed on the X chromosome and frequent unequal recombination and gene conversion may occur between these sequences. X chromosomes may have fusions of the medium- and long-wavelength opsin genes or may have more than one copy of these genes. Defects in this gene are the cause of partial, protanopic colorblindness. [provided by RefSeq, Jul 2008]

Genomic context

Location: Xq28
Sequence: Chromosome: X; NC_000023.10 (153409725..153424507)

See OPN1LW in [Epigenomics](#), [MapViewer](#)

Chromosome X - NC_000023.10

[153285371] [153482764]

MIR718
MECP2
OPN1LW
TEX28P2
OPN1MW
TEX28P1

[www.ncbi.nlm.nih.gov/gene?db=gene&cmd=retrieve&dopt=default&rn=1&list_uids=5956#](#)

Table of contents

- Summary
- Genomic context
- Genomic regions, transcripts, and products
- Bibliography
- Phenotypes
- Interactions
- General gene info
- General protein info
- Reference sequences
- Related sequences
- Additional links

Related information

- Order cDNA clone
- BioAssay
- BioAssays, RNAi Target, Active
- BioProjects
- BioSystems
- Books
- CCDS
- Conserved Domains
- dbVar
- EST
- Full text in PMC
- Genome
- GEO Profiles
- GTR
- HomoloGene
- Map Viewer

OPN1MW on the X Chromosome

OPN1MW opsin 1 (cc x)

www.ncbi.nlm.nih.gov/gene?db=gene&cmd=retrieve&dopt=default&rn=1&list_uids=2652

OPN1MW opsin 1 (cone pigments), medium-wave-sensitive [*Homo sapiens*]

Gene ID: 2652, updated on 2-Feb-2013

Summary

Official Symbol OPN1MW provided by HGNC

Official Full Name opsin 1 (cone pigments), medium-wave-sensitive provided by HGNC

Primary source HGNC:4206

See related Ensembl:ENSG00000147380; HPRD:02365; MIM:300821; Vega:OTTHUMG00000022652

Gene type protein coding

RefSeq status REVIEWED

Organism *Homo sapiens*

Lineage Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorhini; Catarrhini; Hominidae; Homo

Also known as CBD; GCP; GOP; CBBM; COD5; OPN1MW1

Summary This gene encodes for a light absorbing visual pigment of the opsin gene family. The encoded protein is called green cone photopigment or medium-wavelength sensitive opsin. Opsins are G-protein coupled receptors with seven transmembrane domains, an N-terminal extracellular domain, and a C-terminal cytoplasmic domain. The long-wavelength opsin gene and multiple copies of the medium-wavelength opsin gene are tandemly arrayed on the X chromosome and frequent unequal recombination and gene conversion may occur between these sequences. X chromosomes may have fusions of the medium- and long-wavelength opsin genes or may have more than one copy of these genes. Defects in this gene are the cause of deutanopic colorblindness. [provided by RefSeq, Mar 2009]

Genomic context

Location: Xq28

Sequence: Chromosome: X; NC_000023.10 (153448085..153462352)

See OPN1MW in [Epigenomics](#), [MapViewer](#)

Chromosome X - NC_000023.10

[153409725] [153523564]

OPN1LW OPN1MW OPN1MW2

TEX28P2 TEX28P1 TEX28

Related information

- Order cDNA clone
- BioAssay
- BioAssays, RNAi Target, Active
- BioProjects
- BioSystems
- Books
- CCDS
- Conserved Domains
- dbVar
- Full text in PMC
- Genome
- GEO Profiles
- GTR
- HomoloGene
- Map Viewer
- MedGen
- Nucleotide

www.ncbi.nlm.nih.gov/gene?db=gene&cmd=retrieve&dopt=default&rn=1&list_uids=2652#

OPN1SW on Chromosome 7

OPN1SW opsin 1 (cone pigments), short-wave-sensitive [*Homo sapiens*]
Gene ID: 611, updated on 1-Feb-2013

Summary

Official Symbol OPN1SW provided by HGNC
Official Full Name opsin 1 (cone pigments), short-wave-sensitive provided by HGNC
Primary source HGNC:1012
See related [Ensembl:ENSG00000128617](#); [HPRD:01836](#); [MIM:613522](#); [Vega:OTTHUMG00000158311](#)
Gene type protein coding
RefSeq status REVIEWED
Organism [Homo sapiens](#)
Lineage Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi; Mammalia; Eutheria; Euarchontoglires; Primates; Haplorrhini; Catarrhini; Hominidae; Homo
Also known as BCP; BOP; CBT
Summary This gene belongs to the G-protein coupled receptor 1 family, opsin subfamily. It encodes the blue cone pigment gene which is one of three types of cone photoreceptors responsible for normal color vision. Defects in this gene are the cause of tritan color blindness (tritanopia). Affected individuals lack blue and yellow sensory mechanisms while retaining those for red and green. Defective blue vision is characteristic. [provided by RefSeq, Jul 2008]

Genomic context

Location: 7q32.1
Sequence: Chromosome: 7; NC_000007.13 (128412543..128415844, complement)

See OPN1SW in [Epigenomics](#), [MapViewer](#)

Chromosome 7 - NC_000007.13

[128312346] [128462187]

FAM71F2 → FAM71F1 → CALU → OPN1SW → CCDC136 → TRRAP15

Genomic regions, transcripts, and products

[www.ncbi.nlm.nih.gov/gene?db=gene&cmd=retrieve&dopt=default&rn=1&list_uids=611#](#)

Table of contents

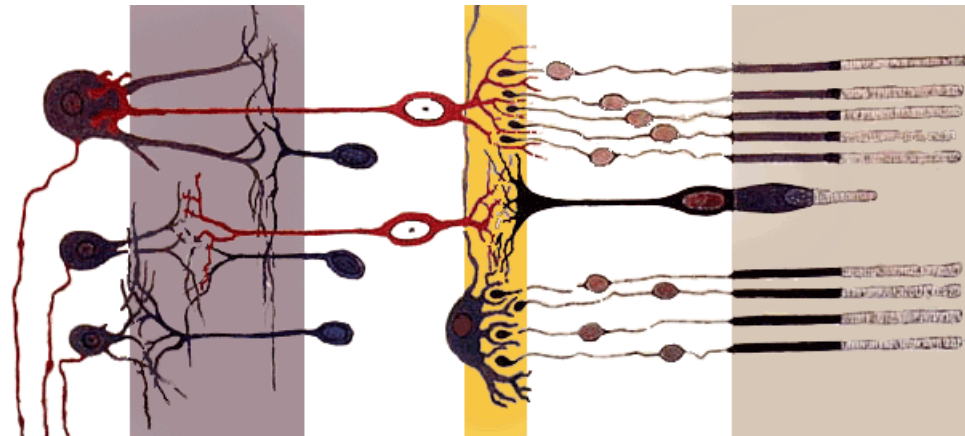
- Summary
- Genomic context
- Genomic regions, transcripts, and products
- Bibliography
- Phenotypes
- General gene info
- General protein info
- Reference sequences
- Related sequences
- Additional links

Related information

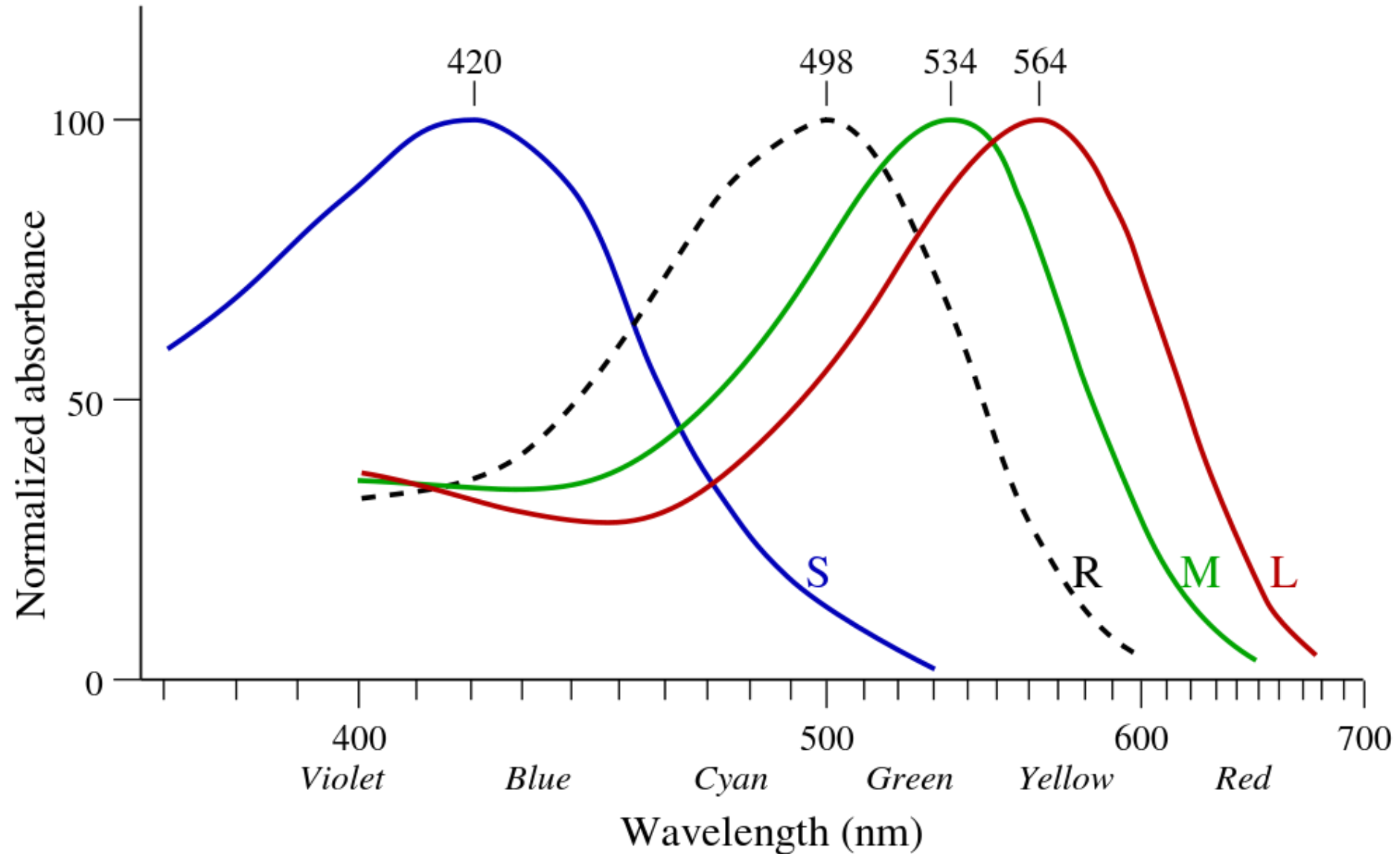
- Order cDNA clone
- BioAssay
- BioProjects
- BioSystems
- CCDS
- Conserved Domains
- dbVar
- EST
- Full text in PMC
- Genome
- GEO Profiles
- HomoloGene
- Map Viewer
- MedGen
- Nucleotide
- OMIM

Rods, Cones and the Retina

- Photoreceptor cells of four kinds in retina
- 7 million cones and 120 million rods
- 1 million optic fibers



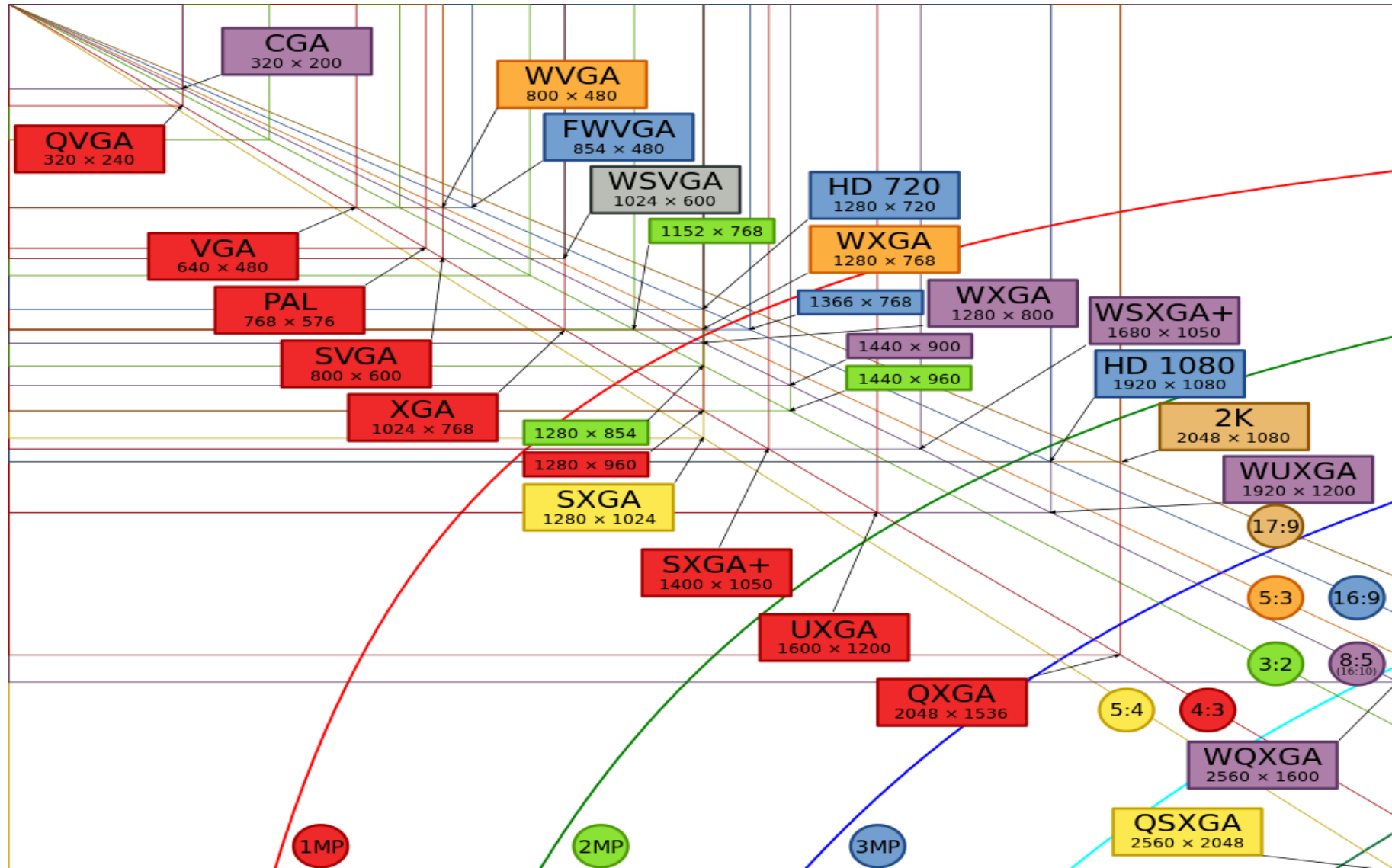
Photoreceptor Absorbance



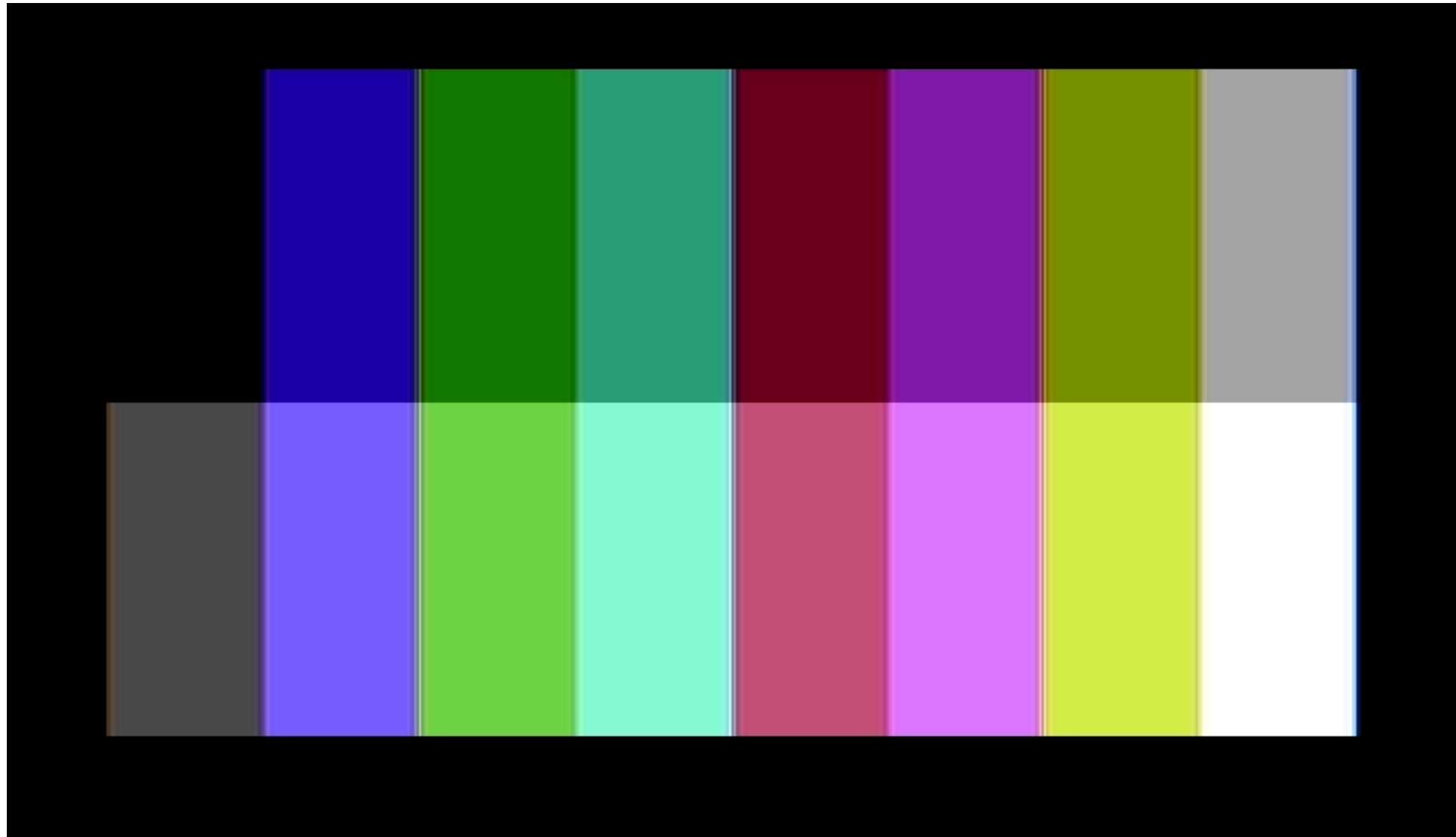
1911 Color Photography with Filters



Computer Display Standards

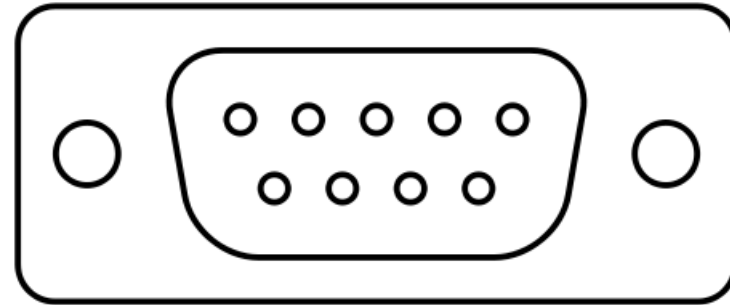


CGA for IBM PC Graphics Card 1981

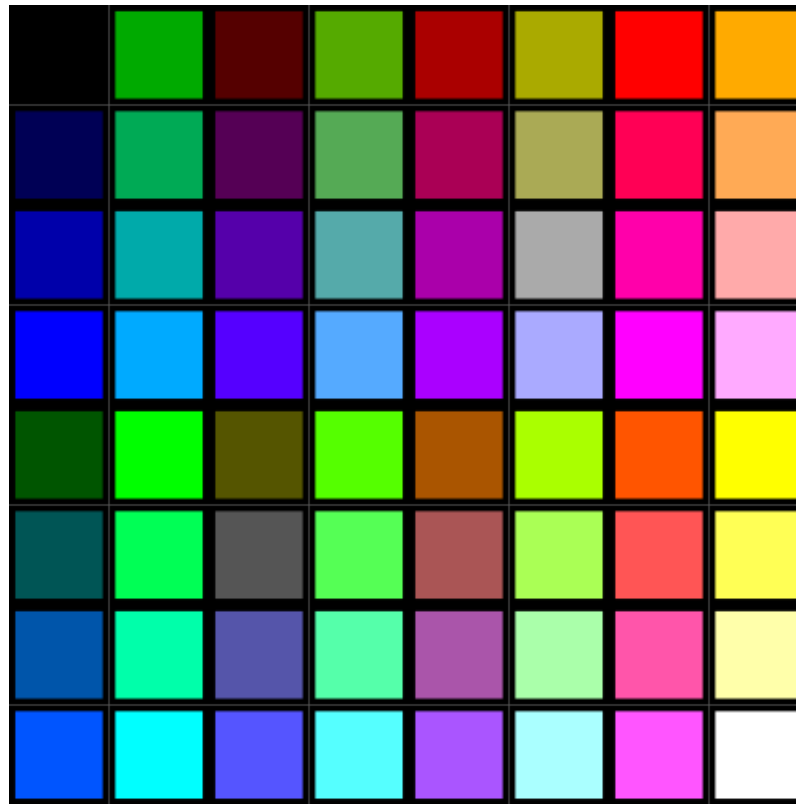


DE-9 Connector for CGA

Pin	Function
1	Ground
2	Ground
3	Red
4	Green
5	Blue
6	Intensity
7	Reserved
8	Horizontal sync
9	Vertical sync

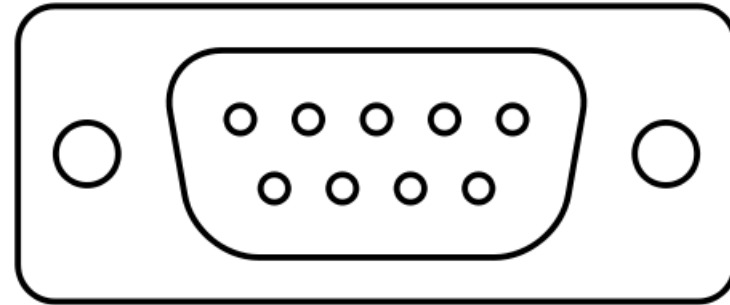


EGA (1984) Has 64 Colors



DE-9 Connector for EGA

Pin	Function
1	Ground
2	Secondary red
3	Red
4	Green
5	Blue
6	Secondary green
7	Secondary blue
8	Horizontal sync
9	Vertical sync

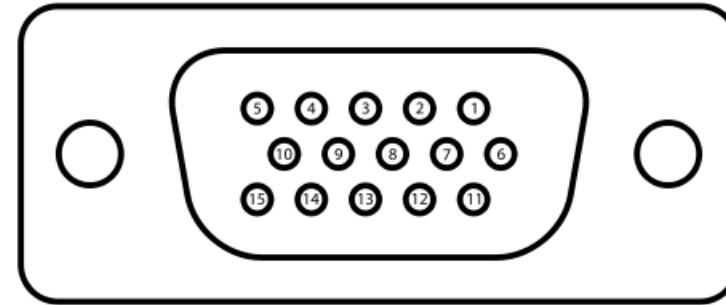


VGA Analog Display (1987)

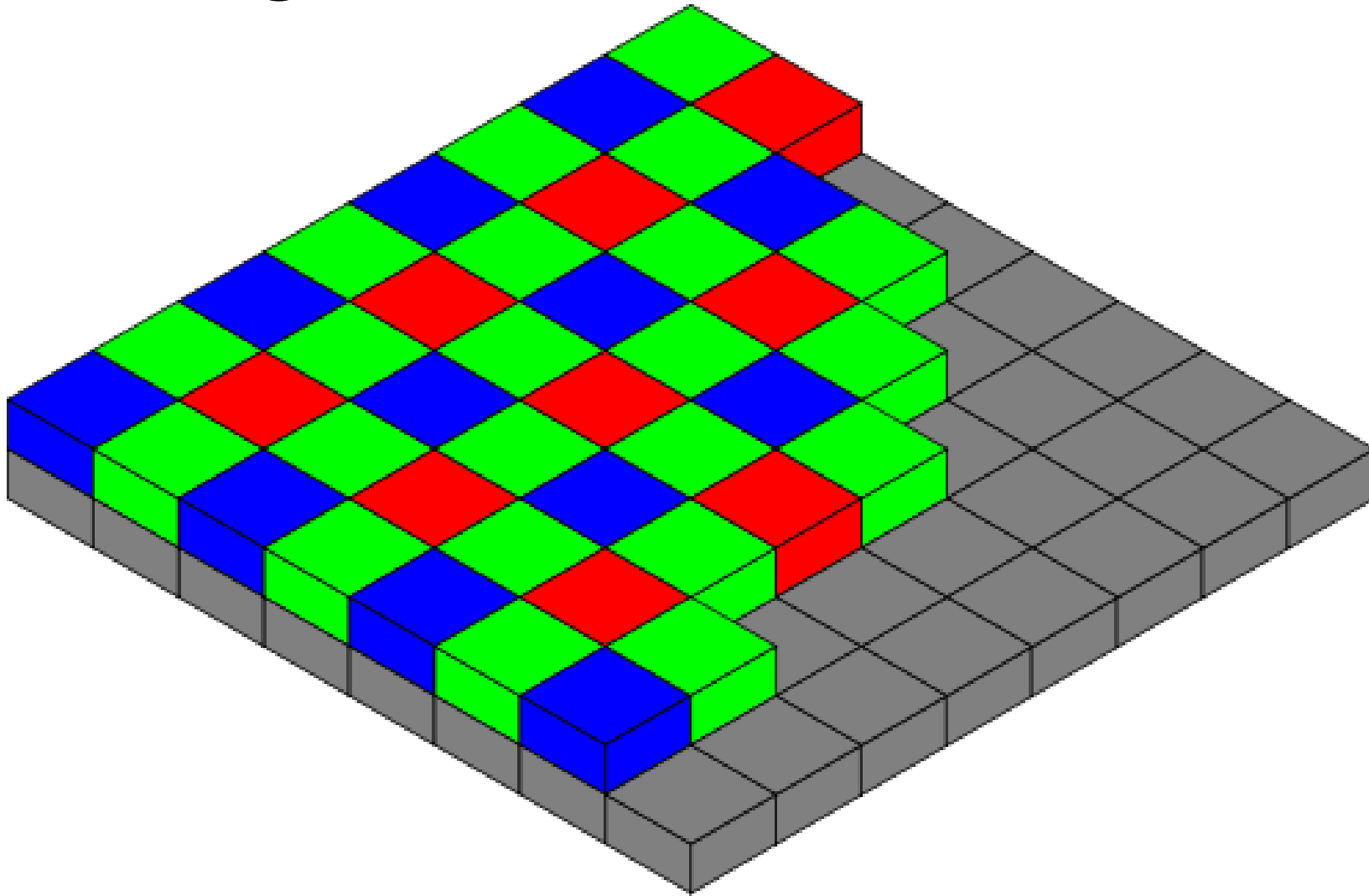
- 640x480
- Analog computer display standard
- 64 possible levels for RGB each.

Pin	Function
1	Red
2	Green
3	Blue
4	Reserved
5	Ground
6	Red return
7	Green return
8	Blue return
9	+5V DC
10	Ground
11	Reserved
12	Reserved
13	Horizontal sync
14	Vertical sync
15	

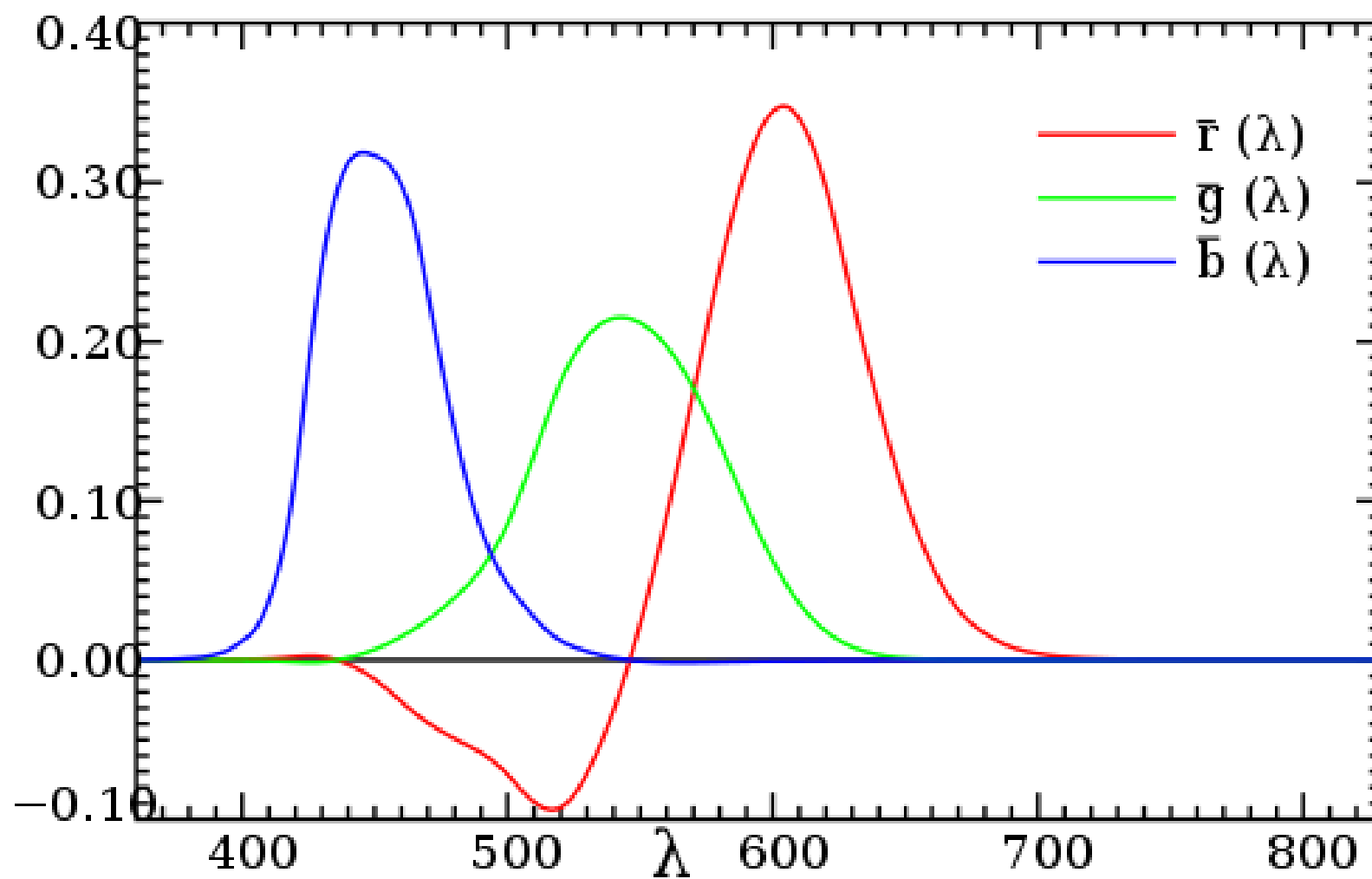
DE15 VGA Connector



Bayer Mask over Charge-Coupled Device (CCD) for Digit Cameras



Tristimulus Values



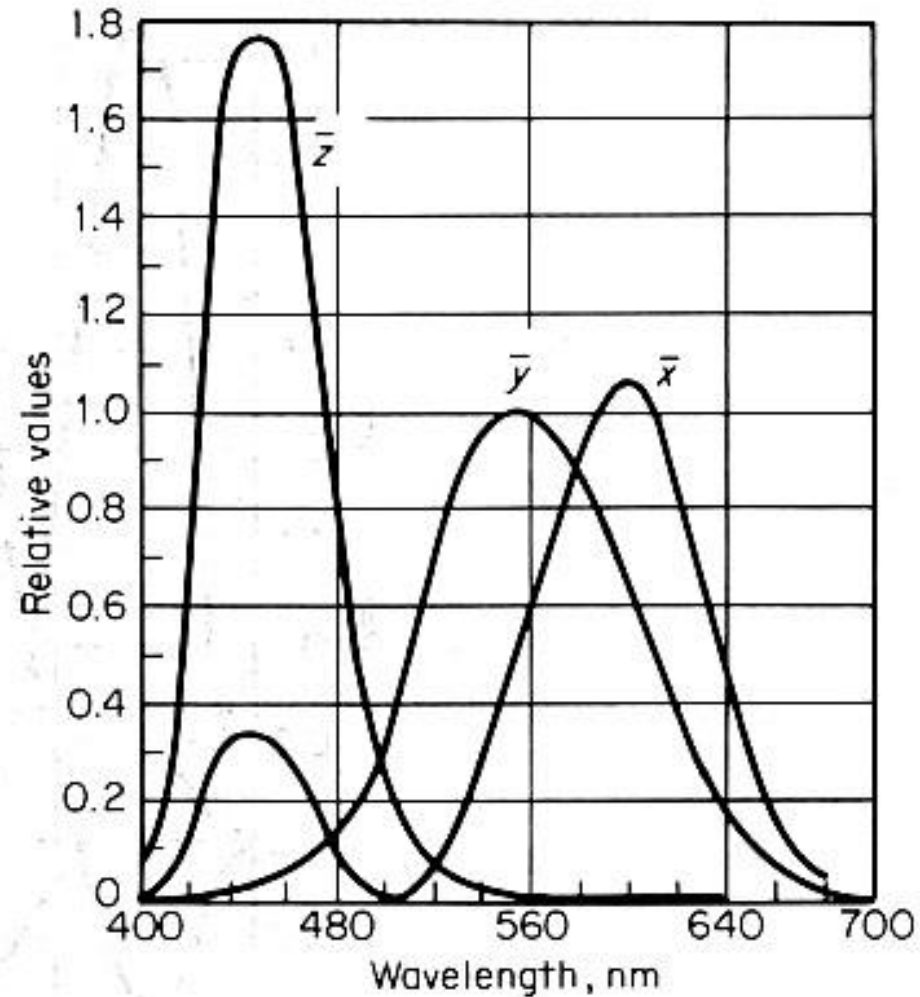
RGB to XYZ in CIE 1931

$$X = 2.7690R + 1.7518G + 1.1300B$$

$$Y = 1.0000R + 4.5907G + 0.0601B$$

$$Z = 0.0000R + 0.0565G + 5.5943B$$

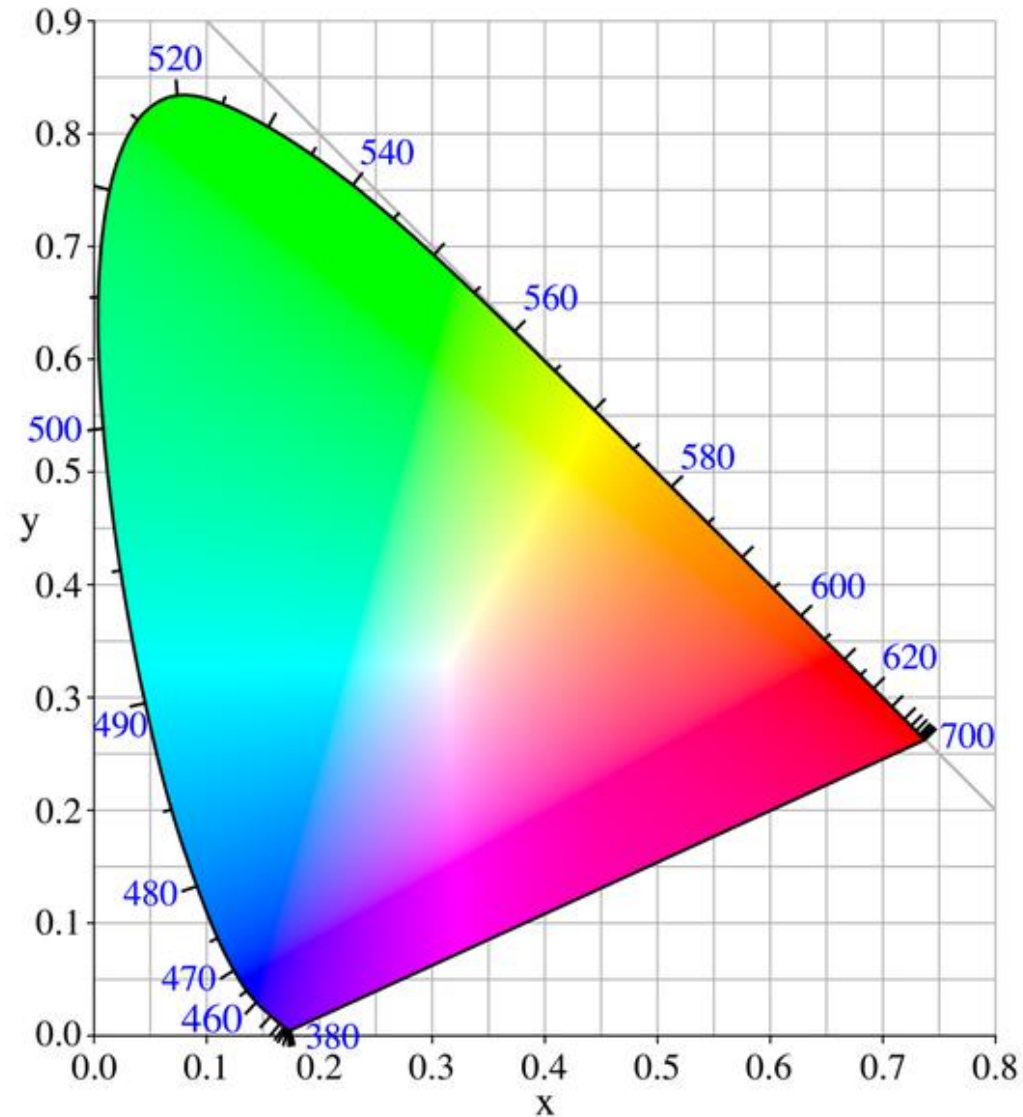
Nonphysical Tristimulus Values



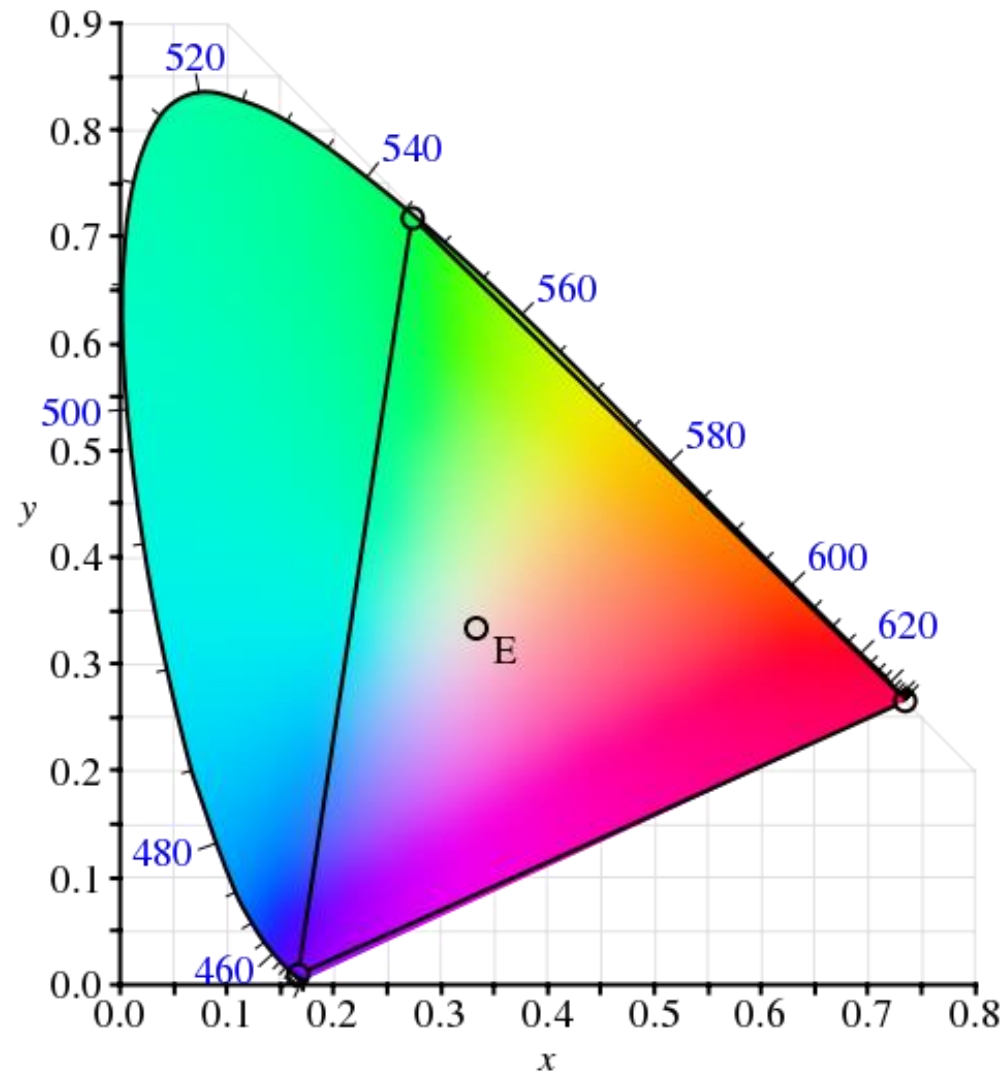
Normalized XYZ

- $x = X / (X + Y + Z)$
- $y = Y / (X + Y + Z)$
- $z = Z / (X + Y + Z) = 1 - x - y$ is redundant

Spectral Locus and Chromaticity



RGB Primaries



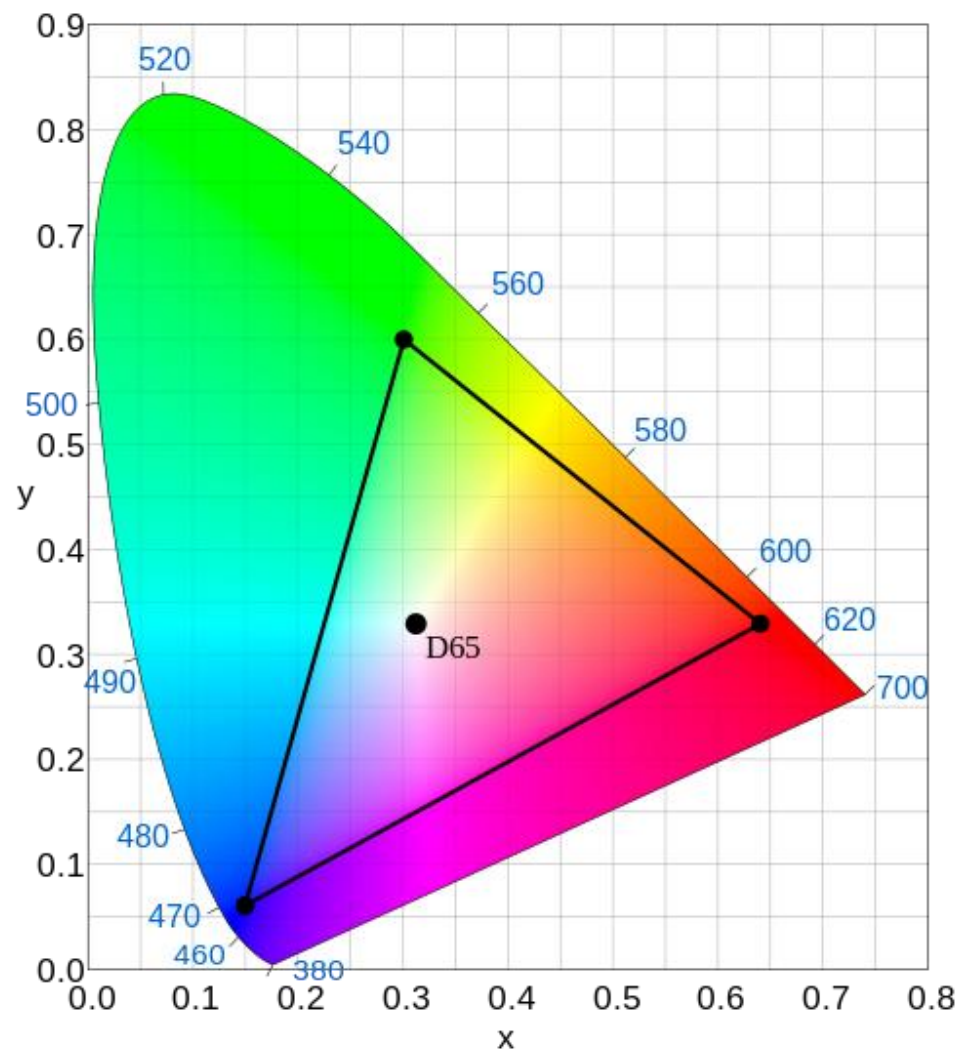
Luminance (ITU-R BT.601, 1982)

- CIE (Commission Internationale de L'Eclairage, International Committee on Illumination) luminance, a characteristic of vision, is the radiant power.
- $Y = (65.738R + 129.057G + 25.064B) / 256 + 16$
- To be compatible to black-and-white TV.

YCbCr Chrominance

- ITU-R BT.601 standard
- Color space transformation
- $Cb = (-37.945R - 74.494G + 112.439B) / 256 + 128$
- $Cr = (112.439R - 94.154G - 18.285B) / 256 + 128$
- The eye is sensitive to small changes in luminance but not in chrominance.
- ITU-R BT.709 is the standard for HDTV.

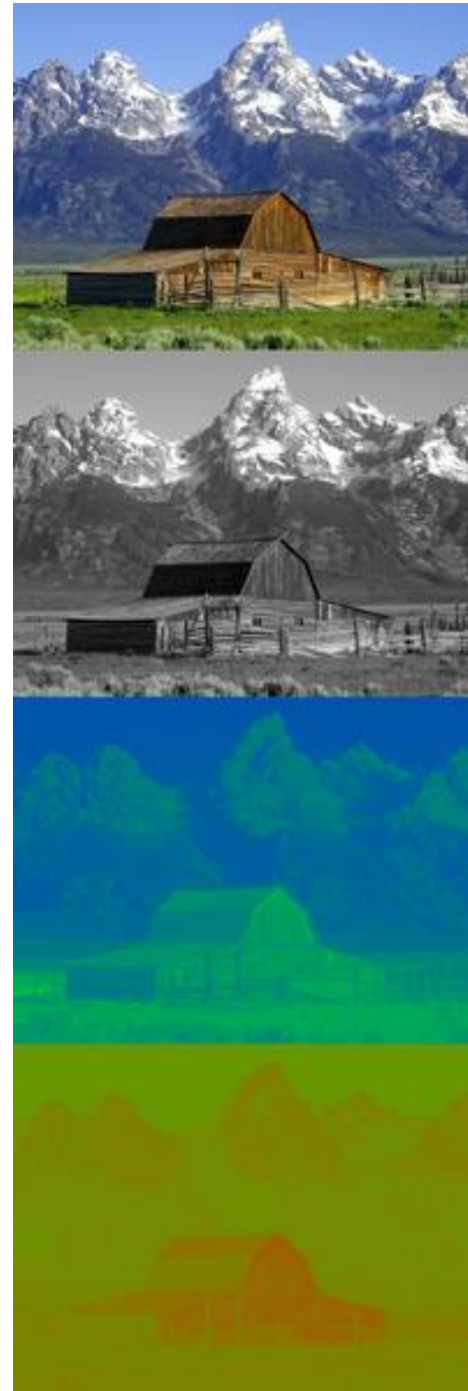
BT 709 (1990) for HDTV



YCbCr (YUV) to RGB

- $R = (298(Y - 16) + 409(Cr - 128))/256$
- $G = (298(Y - 16) - 100(Cb - 128) - 208(Cr - 128))/256$
- $B = (298(Y - 16) + 516(Cb - 128))/256$
- When $Cr = Cb = 128$, $R = G = B = 298(Y - 16)/256$ (grey level)

YUV



YUV420

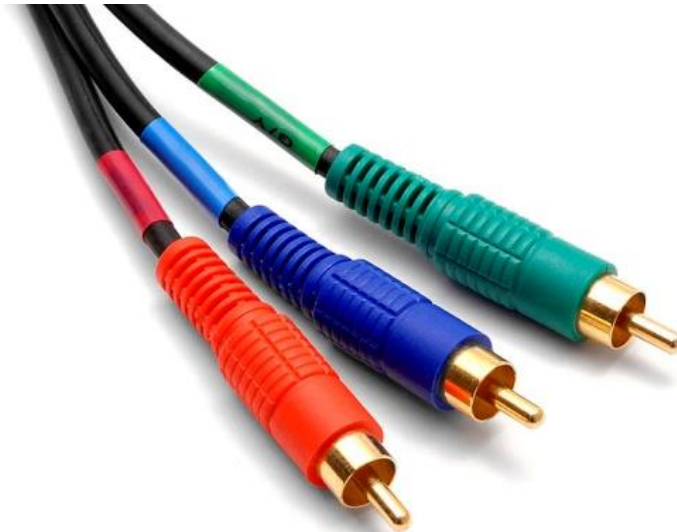
Single Frame YUV420:



Position in byte stream:

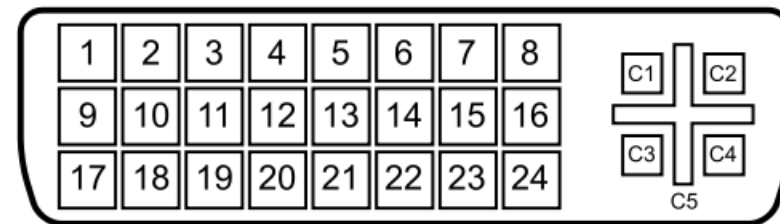


Component Video



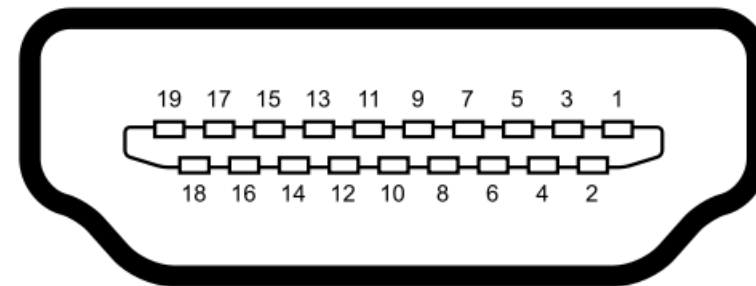
Digital Visual Interface (DVI, 1999)

Pins	Function
1, 2	Red
9, 10	Green
17, 18	Blue
14, 15	Power
6	Clock



HDMI (2002)

Pins	Function
1-3	Data 2
4-6	Data 1
7-9	Data 0
10-12	Clock
18	+5V
19	Hot plug detect



Video Parameters

Application	Frame rate	Resolution	Pixel depth
Surveillance	5	640×480	12
Video telephony	10	320×240	12
Multimedia	15	320×240	16
Analog TV	25	640×480	16
HDTV (720p)	60	1280×720	24
HDTV (1080i)	60	1920×1080	24
HDTV (1080p)	30	1920×1080	24

Table 9.6: Video Parameters for Typical Applications.

Video Resolutions

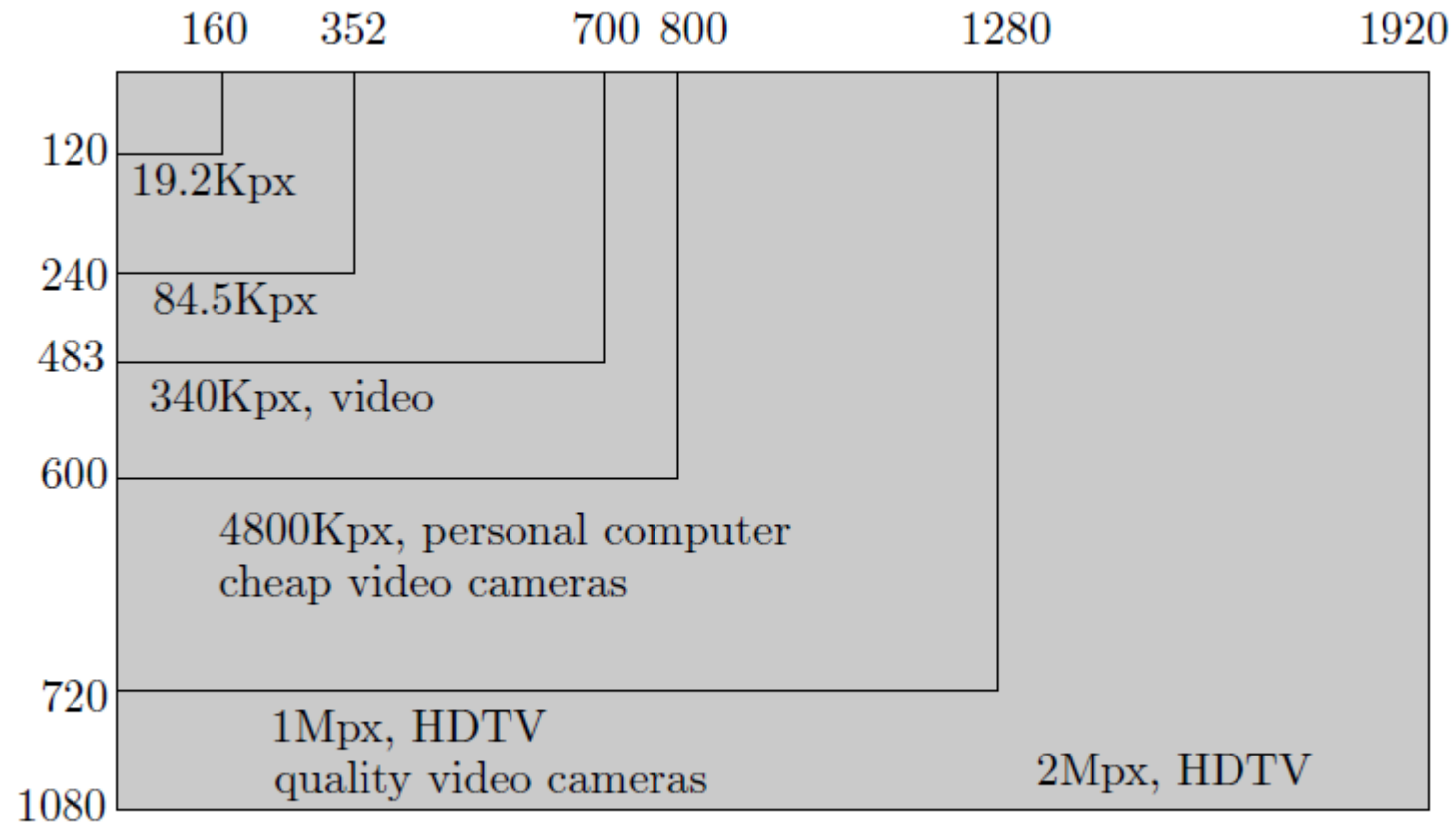


Figure 9.8: Various Video Resolutions.

H.264 or AVC

- H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) is a standard for video compression, and is currently one of the most commonly used formats for the recording, compression, and distribution of high definition video.
- The final drafting work on the first version of the standard was completed in May 2003.

Development of H.264

- H.264/MPEG-4 AVC is a block-oriented motion-compensation-based codec standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC JTC1 Moving Picture Experts Group (MPEG).
- The project partnership effort is known as the Joint Video Team (JVT).
- The ITU-T H.264 standard and the ISO/IEC MPEG-4 AVC standard (formally, ISO/IEC 14496-10 – MPEG-4 Part 10, Advanced Video Coding) are jointly maintained so that they have identical technical content.

H.264 is used in

- High Definition DVDs (Blu-Ray)
- High Definition TV broadcasting in Europe
- Apple products including iTunes video downloads, iPod video and MacOS
- NATO and US DoD video applications
- Mobile TV broadcasting
- Many mobile video services
- Many internet video services
- Videoconferencing
- Consumer camcorders.

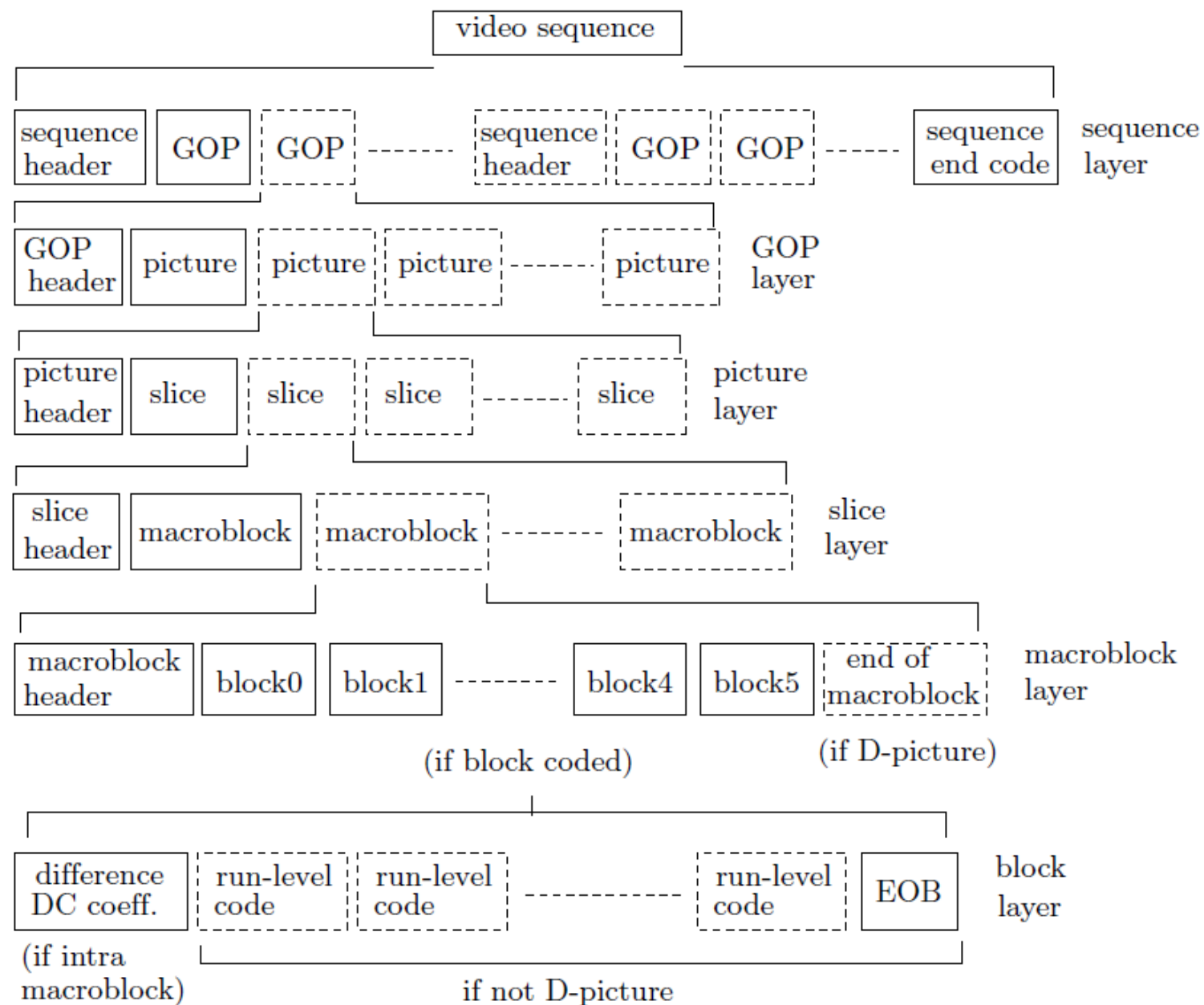
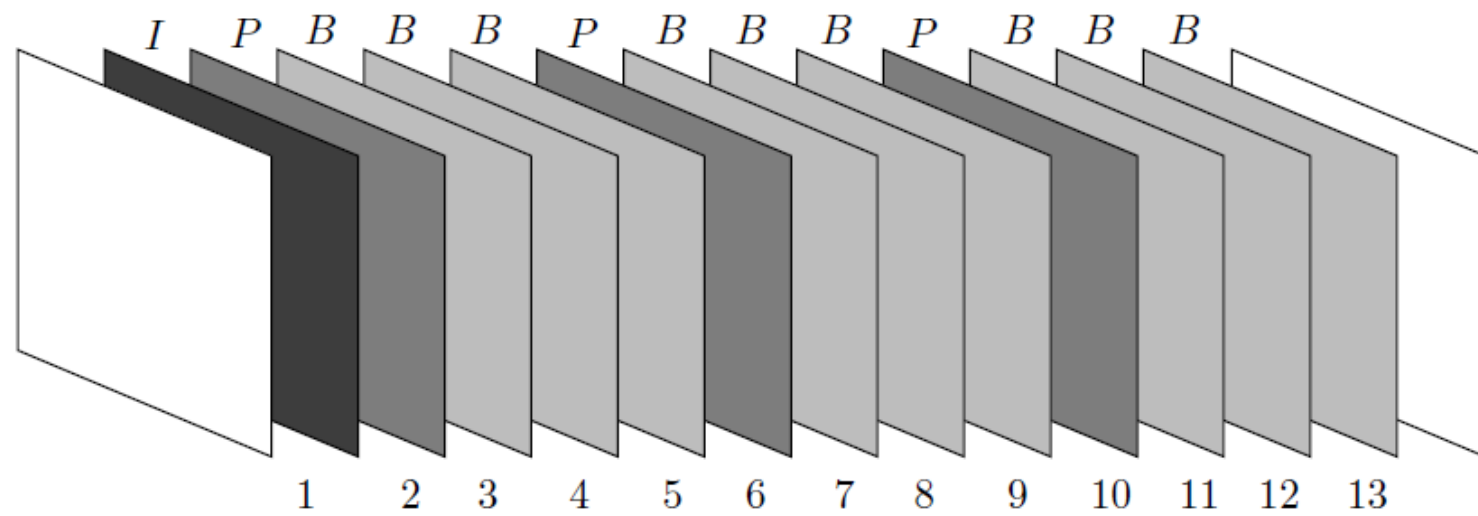
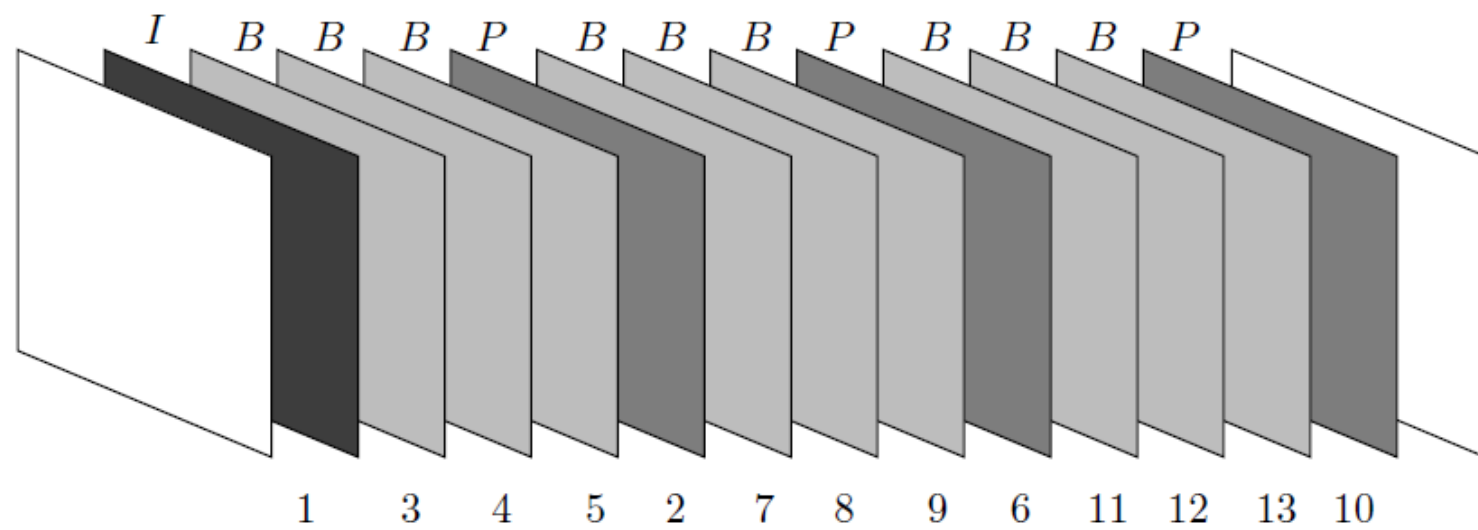


Figure 9.24: The Layers of a Video Stream.



(a)

Time



(b)

Figure 9.9: (a) Coding Order. (b) Display Order.

Motion Compensation

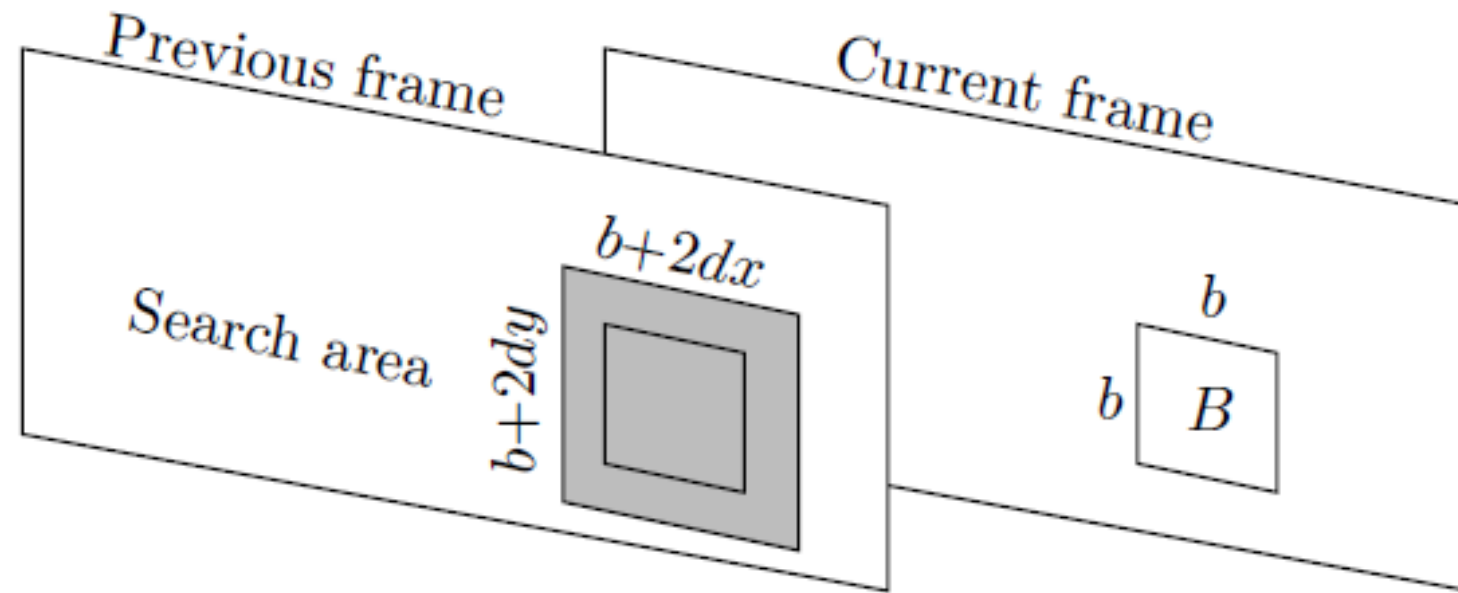


Figure 9.11: Search Area.

H25C: Playback YUV video

```
public class H25C extends JFrame{
    static final int width = 352;
    static final int height = 288;
    static final int halfWidth = width / 2;
    static final int halfHeight = height / 2;
    InputStream in = null;
    int pause = 50;
    int[] pix = new int[height * width];
    int frameSize = height * (width + halfWidth);
    byte[] raw = new byte[frameSize];
    int[][][] yuv = new int[height][width][3];
```

readFrame, raw2yuv, yuv2rgb, drawImage

```
public void playback(){
    Graphics g = getGraphics();
    while (true){
        int len = readFrame();
        if (len < frameSize) return;
        raw2yuv();
        yuv2rgb();
        Image im = createImage(new MemoryImageSource(width,
            height, pix, 0, width));
        g.drawImage(im, 0, 30, null);
        try {
            Thread.sleep(pause);
        } catch (InterruptedException e){}
    }
}
```



```
int readFrame(){
    int len = 0;
    try{
        len = in.read(raw);
    }catch(IOException e){
        System.err.println("IOException");
        System.exit(1);
    }
    return len;
}
```

```
void raw2yuv(){
    int index = 0;
    for (int i = 0; i < height; i++)
        for (int j = 0; j < width; j++){
            yuv[i][j][0] = (raw[index] < 0) ?
                (raw[index] + 256) * 298 : raw[index] * 298;
            index++;
        }
}
```

```

for (int i = 0; i < halfHeight; i++)
    for (int j = 0; j < halfWidth; j++){
        int i2 = i << 1; int j2 = j << 1;
        yuv[i2][j2][1] = yuv[i2][j2 + 1][1] =
        yuv[i2 + 1][j2][1] = yuv[i2 + 1][j2 + 1][1] =
            (raw[index] < 0) ?
                raw[index] + 128 : raw[index] - 128;
        index++;
    }
for (int i = 0; i < halfHeight; i++)
    for (int j = 0; j < halfWidth; j++){
        int i2 = i << 1; int j2 = j << 1;
        yuv[i2][j2][2] = yuv[i2][j2 + 1][2] =
        yuv[i2 + 1][j2][2] = yuv[i2 + 1][j2 + 1][2] =
            (raw[index] < 0) ?
                raw[index] + 128 : raw[index] - 128;
        index++;
    }
}

```

```

void yuv2rgb(){
    int index = 0;
    for (int i = 0; i < height; i++){
        for (int j = 0; j < width; j++){
            int red = yuv[i][j][0] + 409 * yuv[i][j][2] + 128;
            red >>= 8;
            if (red < 0) red = 0; else if (red > 255) red = 255;
            int green = yuv[i][j][0] - 100 * yuv[i][j][1] - 208 * yuv[i][j][2] + 128;
            green >>= 8;
            if (green < 0) green = 0; else if (green > 255) green = 255;
            int blue = yuv[i][j][0] + 516 * yuv[i][j][1] + 128;
            blue >>= 8;
            if (blue < 0) blue = 0; else if (blue > 255) blue = 255;
            pix[index++] = (255 << 24) | (red << 16) |
                (green << 8) | blue;
        }
    }
}

```

H25A.java: video compression

```
static final int width = 352;
static final int height = 288;
static final int numberOfFrames = 200;
static final int frameSize = width * height * 3 / 2;
byte[] buffer = new byte[frameSize * numberOfFrames];
byte[] result = new byte[numberOfFrames * frameSize];
int compressedDataLength = 0;

void readFrames(){
    try{
        System.in.read(buffer);
    }catch(IOException e){
        System.err.println("IOException");
        System.exit(1);
    }
}
```

Simple Prediction Using Previous Frame

```
void differential(){
    for (int frame = numberOfFrames - 1; frame > 0; frame--){
        int frameBase = frame * frameSize;
        for (int j = 0; j < frameSize; j++){
            int v = buffer[frameBase + j];
            int u = buffer[frameBase - frameSize + j];
            int diff = v - u;
            buffer[frameBase + j] = (byte)diff;
        }
    }
}
```

Compression Using Deflater

```
void compress(){  
    Deflater compressor = new Deflater();  
    compressor.setInput(buffer);  
    compressor.finish();  
    compressedDataLength = compressor.deflate(result);  
    System.out.write(result, 0, compressedDataLength);  
    System.out.flush();  
}
```

```
void readCompressedData(){
    try{
        compressedDataLength = in.read(result);
    }catch(IOException e){
        System.err.println("IOException");
        System.exit(1);
    }
}
```

```
void decompress(){
    try {
        Inflater decompressor = new Inflater();
        decompressor.setInput(result, 0, compressedDataLength);
        int resultLength = decompressor.inflate(buffer);
        decompressor.end();
        if (resultLength != numberOfFrames * frameSize){
            System.err.println(resultLength);
            System.exit(1);
        }
    } catch (java.util.zip.DataFormatException ex) {
        System.err.println(ex.getMessage());
        System.exit(1);
    }
}
```

H25B.java: Video Decompression and Playback

```
public void playback(){
    Graphics g = getGraphics();
    for (int i = 0; i < numberOfFrames; i++){
        getFrame(i);
        raw2yuv();
        yuv2rgb();
        Image im = createImage(new MemoryImageSource(width,
            height, pix, 0, width));
        g.drawImage(im, 0, 30, null);
        try {
            Thread.sleep(pause);
        } catch (InterruptedException e){}
    }
}
```



```
void getFrame(int frame){
    int frameBase = frame * frameSize;
    if (frame > 0)
        for (int j = 0; j < frameSize; j++){
/* This is how we got buffer[frameBase + j} in H25A.java:
```

```
    int v = buffer[frameBase + j];
    int u = buffer[frameBase - frameSize + j];
    int diff = v - u;
    buffer[frameBase + j] = (byte)diff;
```

We need its inverse.

Given the recovered preceding frame value u and the differential
in buffer[frameBase + j], write the code to replace buffer[frameBase + j}
with the recovered v.

```
*/
    }
    for (int i = 0; i < frameSize; i++) raw[i] = buffer[frameBase + i];
}
```

Homework 25: due 4-27-15

- Complete the `getFrame()` function in `H25B.java` and run it on `test25`.