Psychtoolbox编程1

蔡永春 副教授 博导
浙江大学 心理与行为科学系
yccaicourse@163.com
18857875045

2023-11-8日报告文献

- 1、Duje Tadin, Joseph S. Lappin, Lee A. Gilroy & Randolph Blake (2003). Perceptual consequences of centre–surround antagonism in visual motion processing. Nature, 424:312-315.
- 2、补充材料： Duje Tadin. (2004). Perceptual consequences of centre–surround antagonism in visual motion processing. Dissertation of Tadin, Chapter 3.

期限：2023年11月5日晚24点前提交

提交方式：上传至**学在浙大**

文件命名格式：姓名. doc

作业要求

1、提交阅读报告：

- 用1000-1500字介绍所阅读的论文（研究背景、研究方法、主要发现等）；
- 你对文章所研究问题的理解或思考（你对本研究结果的理解、本研究对你的启发、基于此研究能否提出新的研究问题等）；

2、课堂报告（1位同学）：

- 不超过30分钟；
- 自由报名；
- 报告加分：本次作业加5-15分；如本已报名或被抽中报告但没有准备的，本次作业记0分；

目 录
Outline

- 心理学编程需求及PTB
- 刺激的表达及显示原理
- 呈现静态图
- 呈现动态图

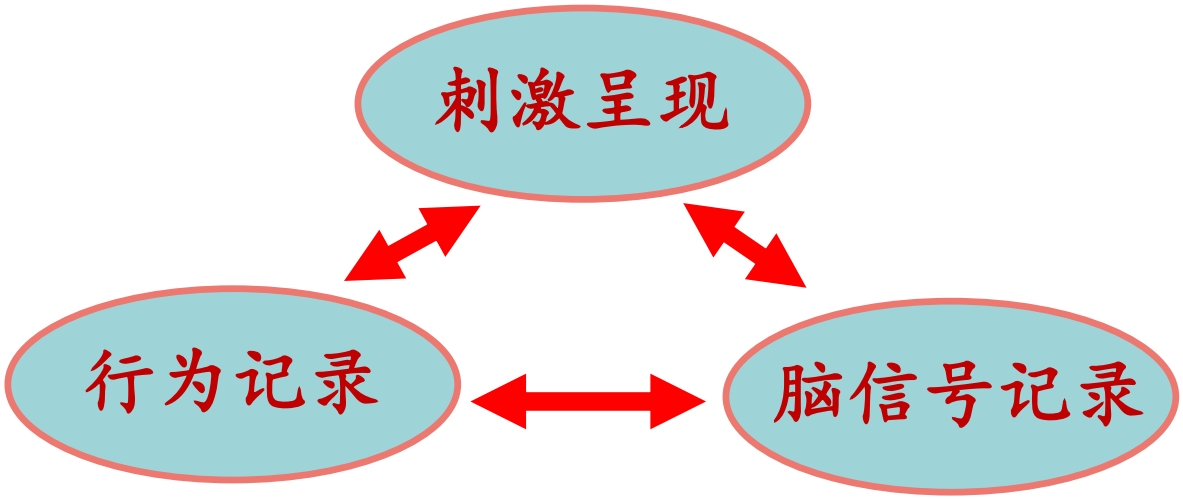
目 录

Outline

- 心理学编程需求及PTB
- 刺激的表达及显示原理
- 呈现静态图
- 呈现动态图

心理学编程需求及PTB介绍

- （认知）心理学研究特点及对编程语言的要求



- 实验刺激：对刺激准确、灵活(时间、亮度)的控制；
- 被试反应：对被试反应高精度（时间）的记录；
- 多设备的同步：准确、容易实现；

什么样的编程语言适合用于心理学编程？

- 低级编程语言：

  对硬件的控制精确，难学（如汇编 C/C++等）；

- 高级编程工具：

  易学，对硬件的控制差，灵活度低（如Eprime）；

- 中级编程语言：

  易学、灵活，通过与低级语言混合编程，可实现对硬件的精确控制（如Matlab-PTB, Python）；

- Psychtoolbox3

  由一系列的Matlab函数（命令）构成，可方便/精确/灵活地生成各种是视觉/听觉刺激、记录被试反应，被广泛地应用于心理学、神经科学领域。

- C语言编写的可执行文件(mex文件)用于控制视觉刺激/听觉刺激，可与显卡、声卡等硬件直接交互，这保证了心理学刺激的精确性；
- 用Matlab语言编写的m函数命令大大简化了心理学数据处理，例如QUEST工具包、屏幕矫正程序等；

- PTB1: 1995, by David Brainard, only support Macintosh ;
- PTB2: 2000, by David Brainard, Denis Pelli and Allen Ingling and others, support MacOS and Windows ;
- PTB3: 2006, by Mario Kleiner, Richard Murray, Tobias Wolf & many others, support MacOS, Windows and Linus ;

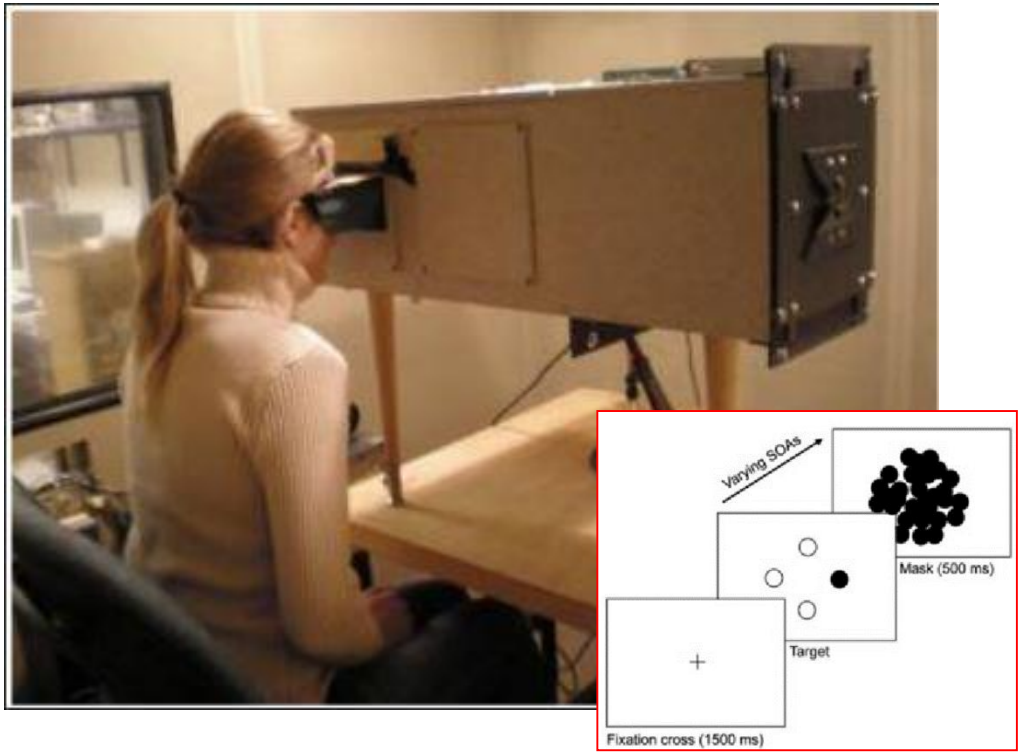
PTB主页：http://psychtoolbox.org/

常见问题： https://github.com/Psychtoolbox-3/Psychtoolbox-3/wiki/FAQ

PTB编程实例：http://peterscarfe.com/ptbtutorials.html

Totally free !

早期的视觉刺激生成设备



tachistoscope（速视仪）

Cambridge Research System
约20万元人民币

目 录

Outline

- 心理学编程需求及PTB
- 刺激的表达及显示原理
- 呈现静态图
- 呈现动态图

刺激的表达及显示原理

- 最简单的PTB程序-----新建窗口

  所有PTB编程的第一步，建立一个刺激呈现窗口，所有的视觉刺激都呈现于这个窗口之内。

```
%%Example 1-1: 新建一个PTB窗口
ScreenNumber=0;
WinColor=[100,100,100];
[WindowPtr,rect]=Screen('OpenWindow',ScreenNumber,WinColor);
WaitSecs(5);%等待5秒
Screen('CloseAll');%关闭窗口
```

[windowPtr, Rect]=Screen('OpenWindow', ScreenNumber, Color, Rect, PixelSize, NumberOfBuffers, Stereomode, Multisample, Imagingmode);

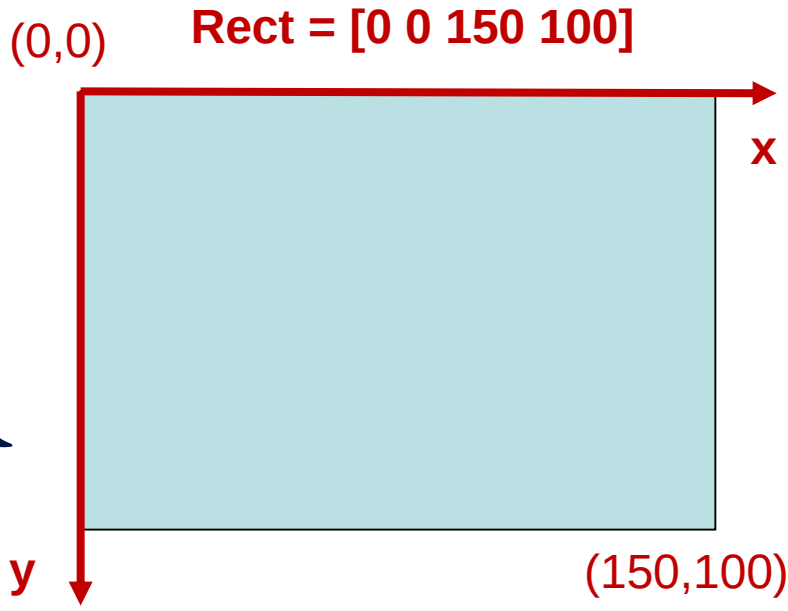
某些参数可缺省

[windowPtr,rect]=Screen('OpenWindow', ScreenNumber [,color]);

Matlab-PTB3帮助：Screen('OpenWindow?')

windowPtr：窗口编号；

Rect：窗口的大小；



ScreenNumber：屏幕编号，0、1、2；

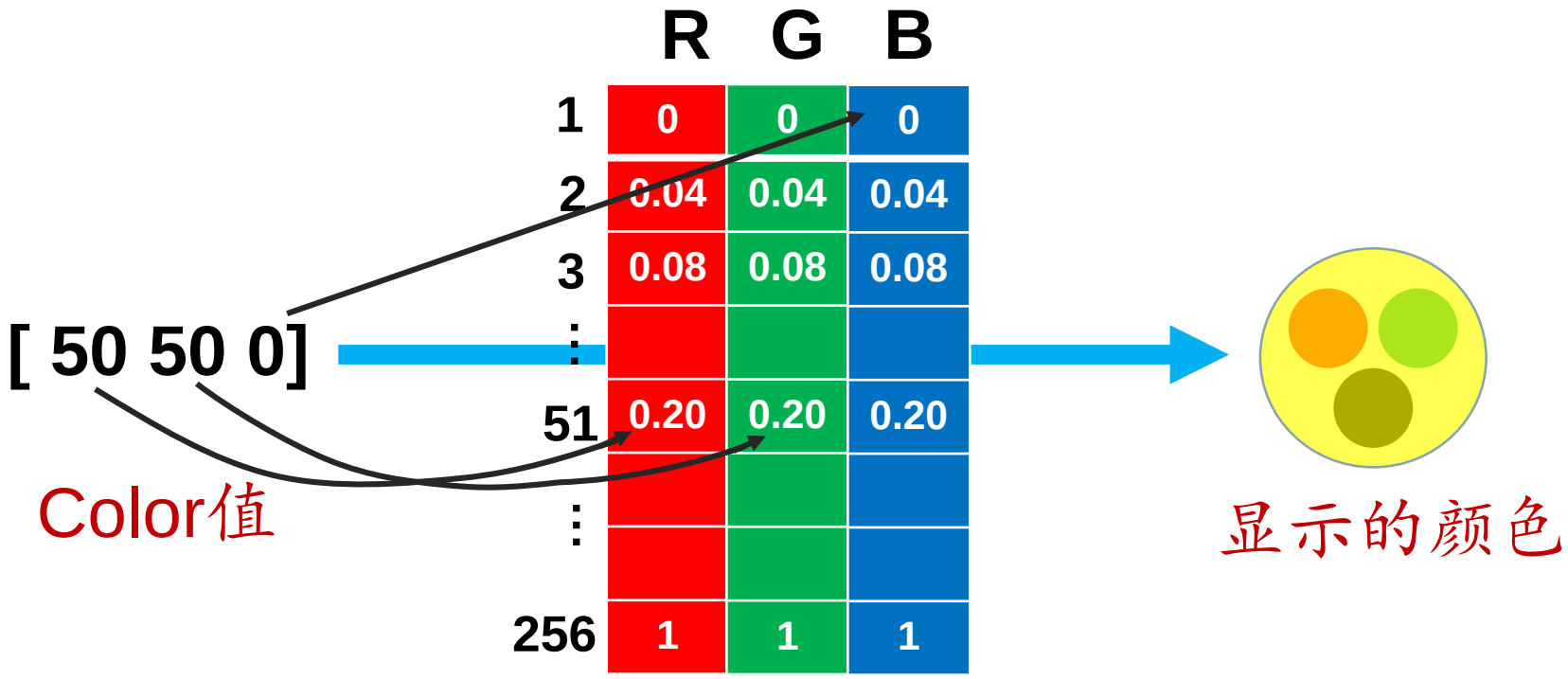
Color：窗口的颜色 [0,0,0] - [255,255,255]

PixelSize：窗口内每个像素的位数，默认值为32，[R G B]各8位+8位alpha（透明度）。

- Color值如何转化为屏幕上显示的颜色？

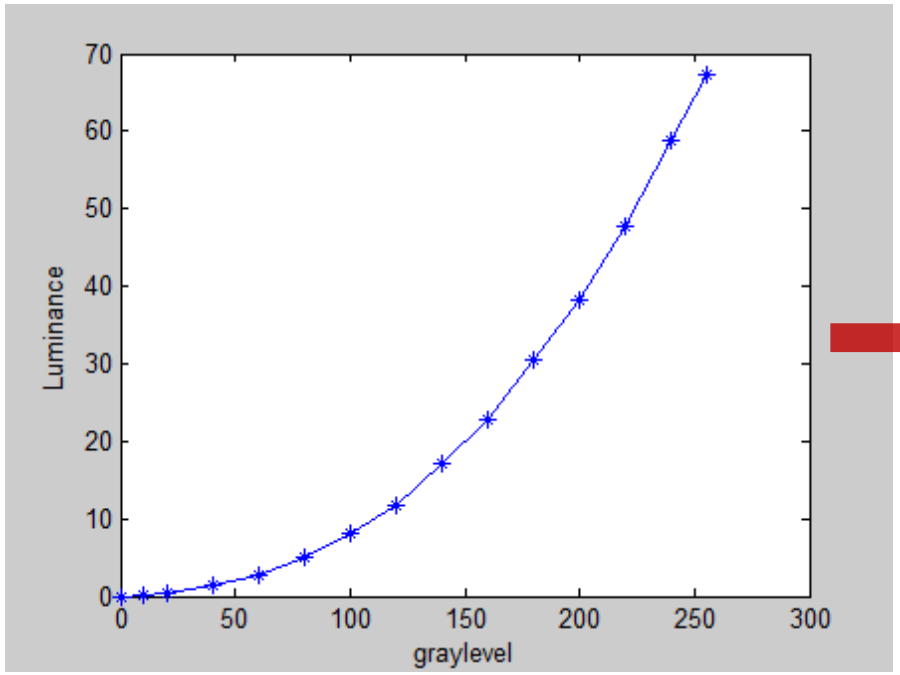
Color = [R G B]

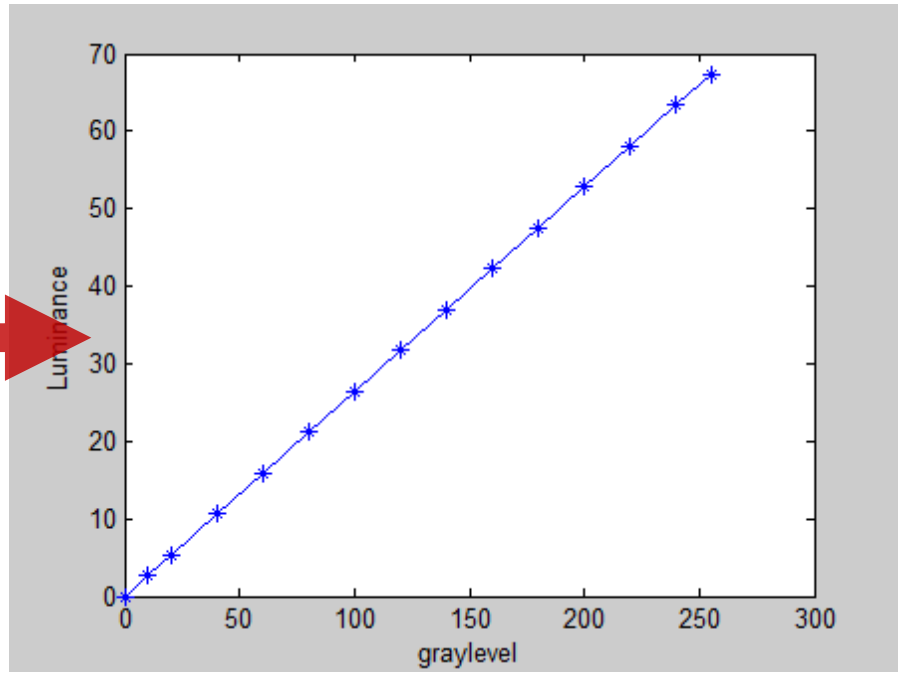
RGB分别表示每个像素点红、绿、蓝三种颜色的强度

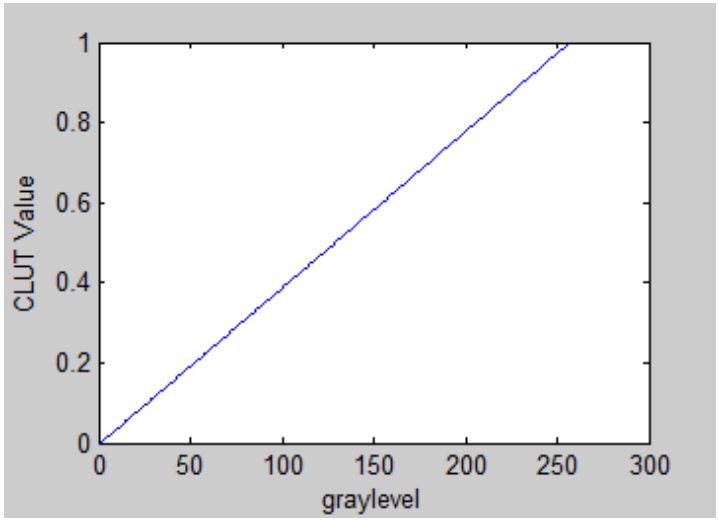


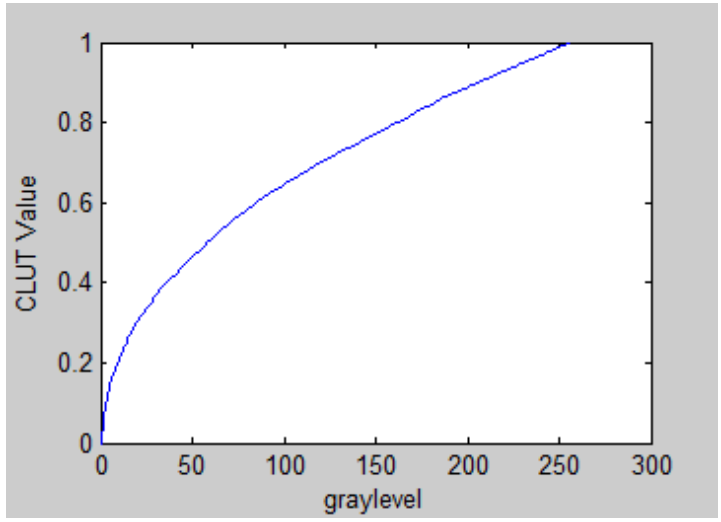
Color值 → ColorLookUpTable(显卡电压值) → 亮度值

- 通过改变CLUT值实现屏幕线性校正









画几何图形（矩形、圆、多边形、线段、弧线等）

Screen(‘FillRect’, windowPtr [,color] [,rect]);

Screen('FillOval', windowPtr [,color] [,rect]);

Screen('FrameRect', windowPtr [,color] [,rect] [,penWidth]);

Screen('FrameOval', windowPtr [,color] [,rect] [,penWidth] …
[,penHeight] [,penMode]);

Screen('DrawLine', windowPtr [,color], fromH, fromV, toH, toV [,penWidth]);

Screen('DrawArc',windowPtr,[color],[rect],startAngle,arcAngle)

Screen(‘FrameArc’,windowPtr,[color],[rect],startAngle,arcAngle[,penWidth] …
[,penHeight] [,penMode])

Screen('FillArc',windowPtr,[color],[rect],startAngle,arcAngle)

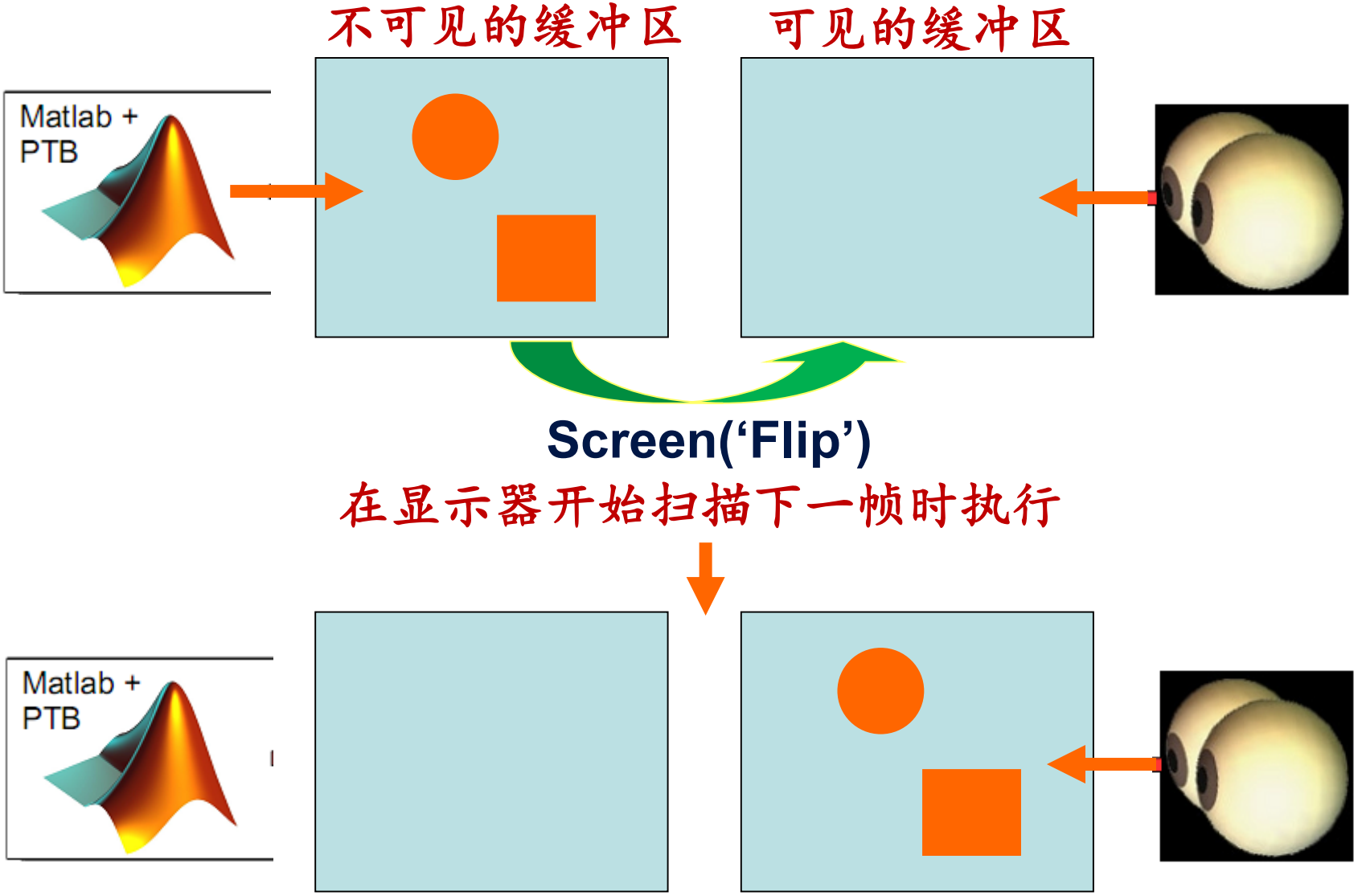
Screen('FramePoly', windowPtr [,color], pointList [,penWidth]);

Screen('FillPoly', windowPtr [,color], pointList);

```
%%example1-2 画一个实心矩形和一个空心圆
WindowNumber=0;
WinColor=[100,100,100];
[WindowPtr,Rect]=Screen('OpenWindow',WindowNumber,WinColor);
Screen(‘FillRect’, WindowPtr,[255,0,0],[100,100,200,200]);%画矩形
Screen(‘FrameOval’,WindowPtr,[0,0,255],[300 300 500 500]);%画圆
Screen('flip',WindowPtr);%呈现
WaitSecs(5);%等待5秒
Screen('CloseAll');%关闭所有窗口
```

关键命令Screen(‘Flip’)：把已经画好的刺激呈现到当前窗口

双缓冲技术(Double Buffering)



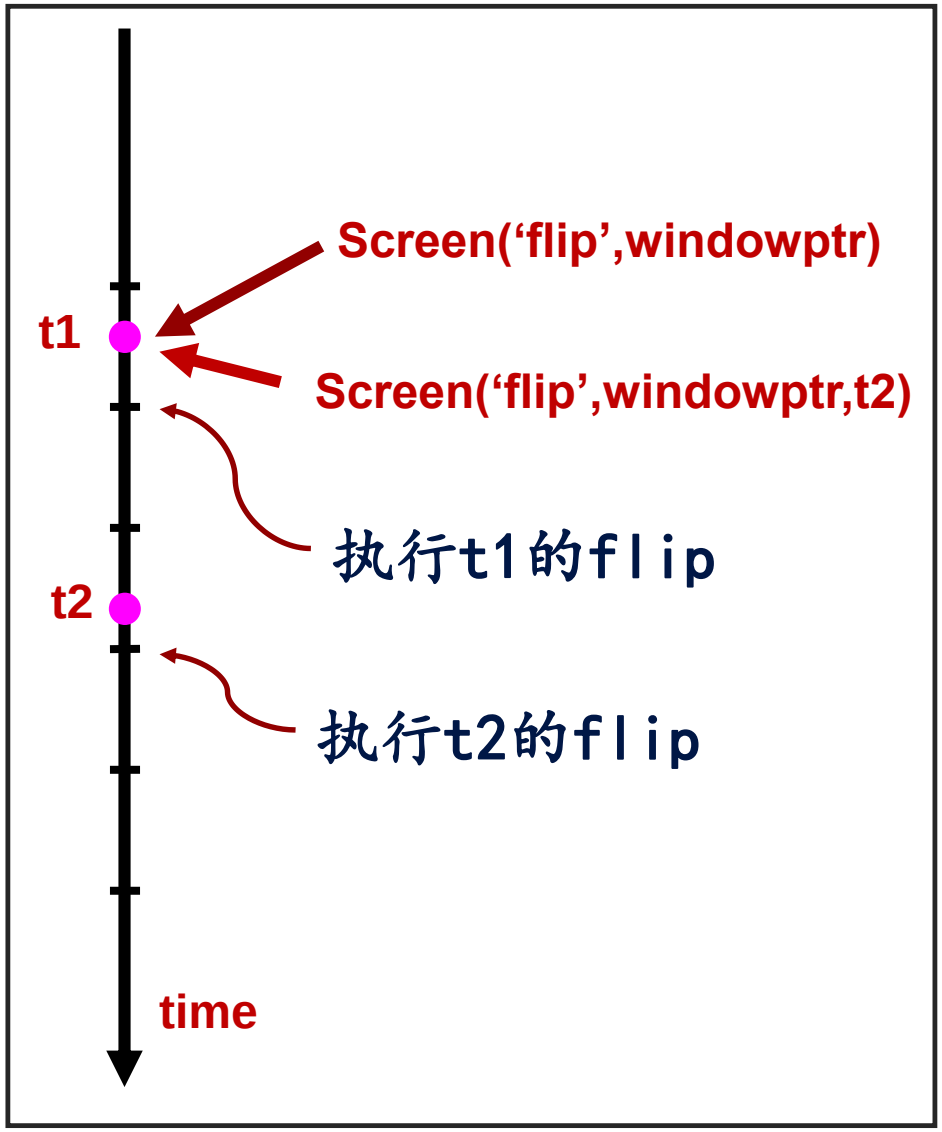
双缓冲技术实现了边画刺激边呈现，是PTB的一个重大改进。

- Screen(‘flip’) 命令

[VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] = Screen(‘Flip’, windowPtr [, when])

windowPtr: 窗口编号

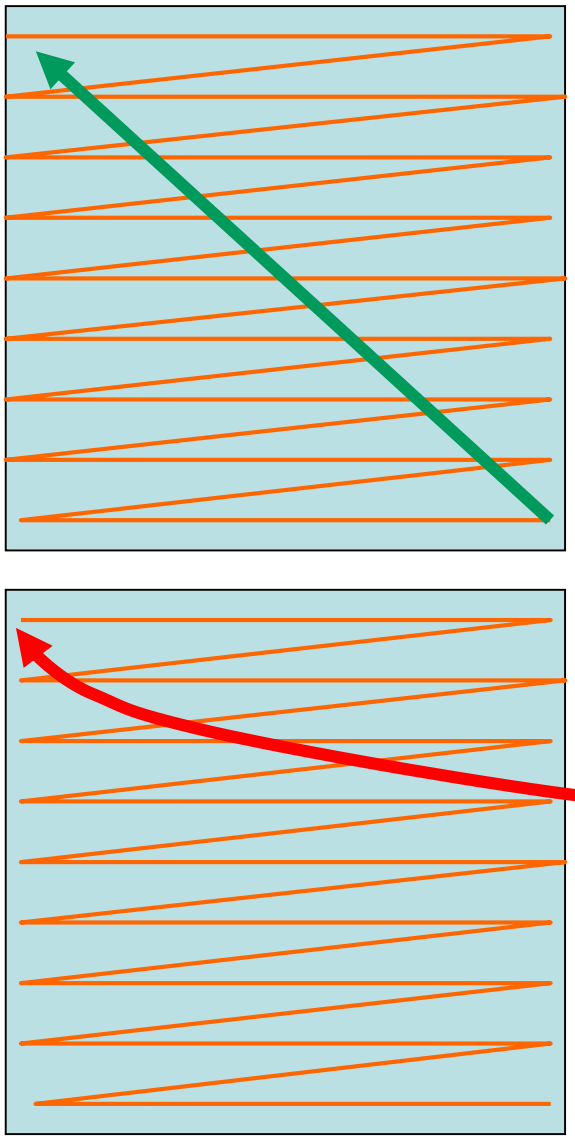
when: 执行flip的时刻



- 显示设备

CRT显示器比LCD显示器拥有更高的时间精度和颜色精度，更适合用作视觉实验显示设备。

[VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =
Screen('Flip', windowPtr [, when])



PTB工作，画图、检测按键等。

VBLTimestamp,双缓冲切换时间。回扫(Vertical blank)，PTB休眠，OS工作。

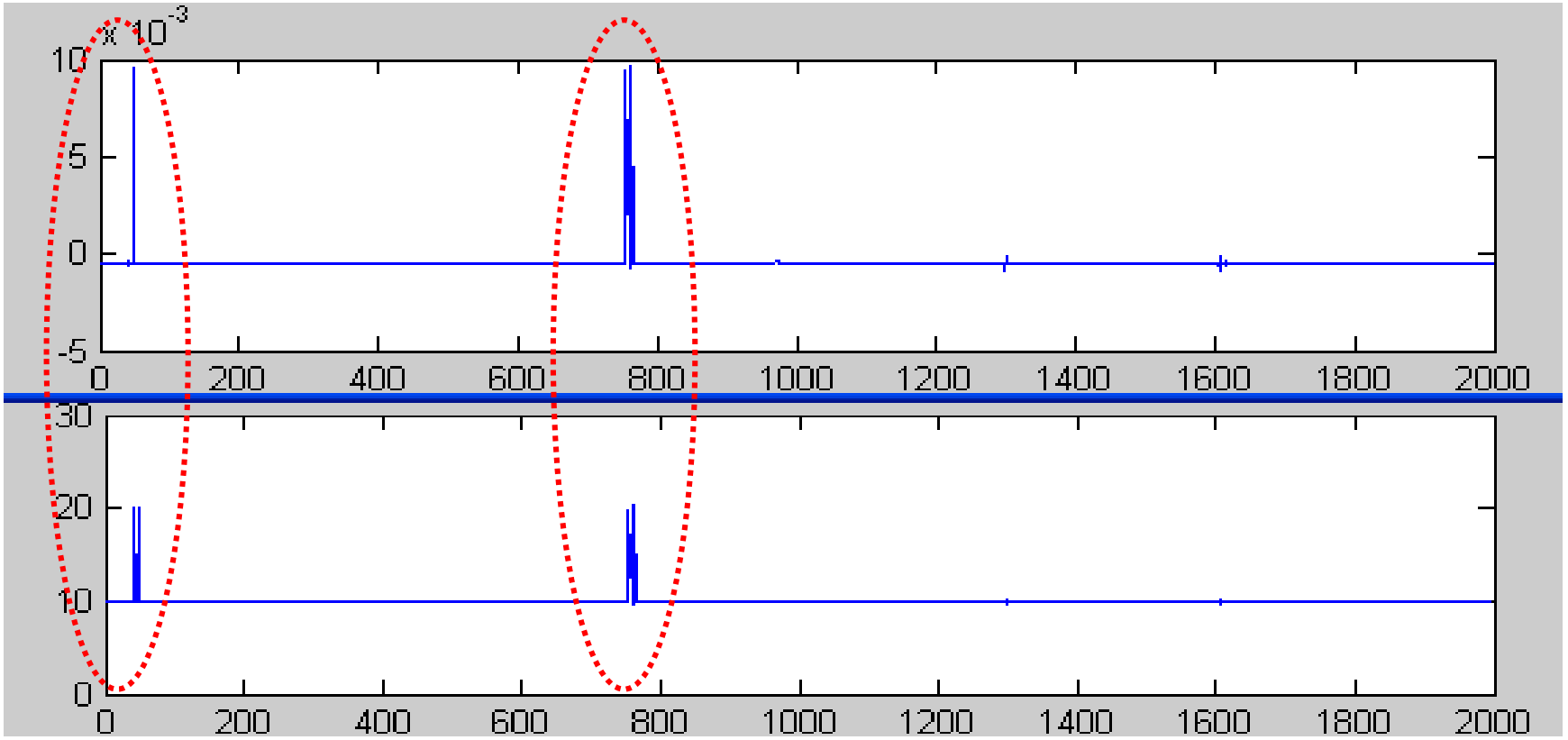
FlipTimestamp, PTB唤醒时间，可能会有波动。

StimulusOnsetTime =
FlipTimestamp - T晚醒

T晚醒, 可由Beamposition推算得出。

Missed >0表明此帧被错过; (可能不准)

- 参数Missed可用来监测是否有丢帧，但时常有误报。

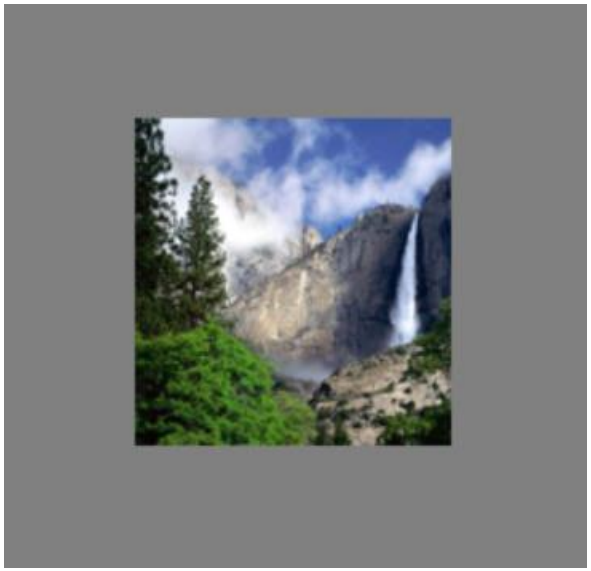


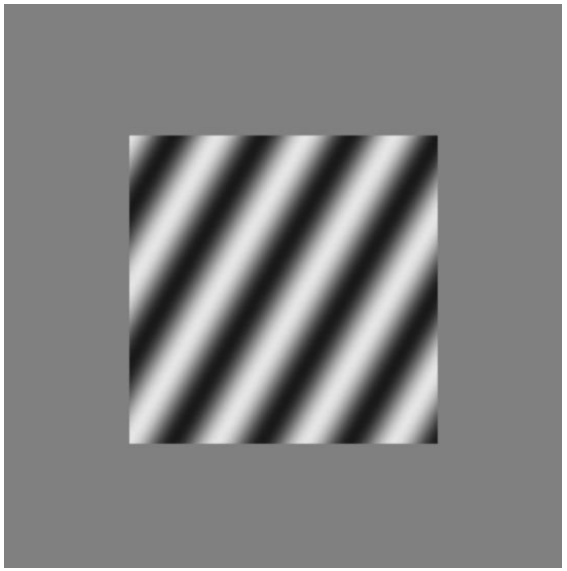
目 录

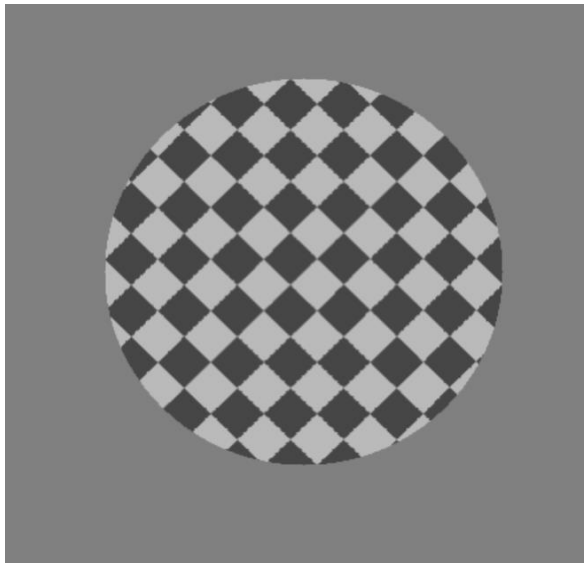
Outline

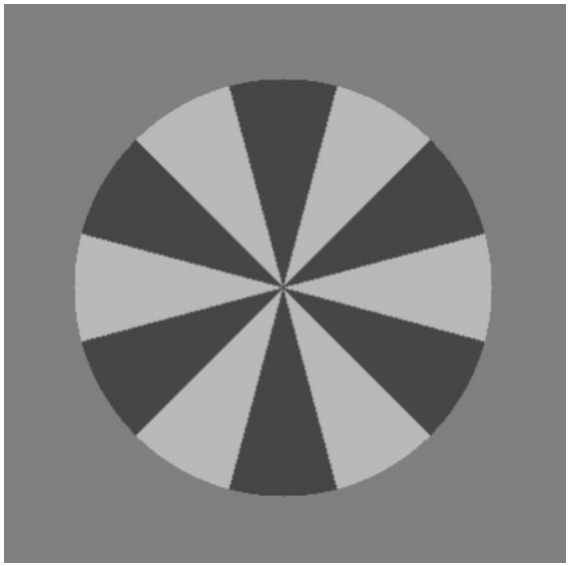
- 心理学编程需求及PTB
- 刺激的表达及显示原理
- 呈现静态图
- 呈现动态图

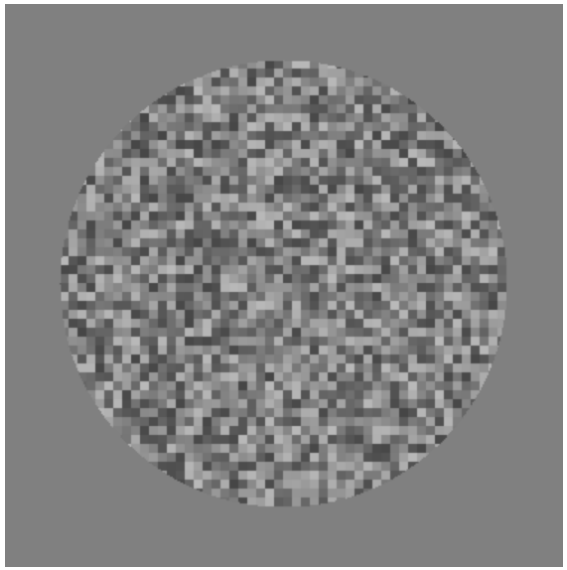
如何生成心理(物理)学实验中的常见静态刺激











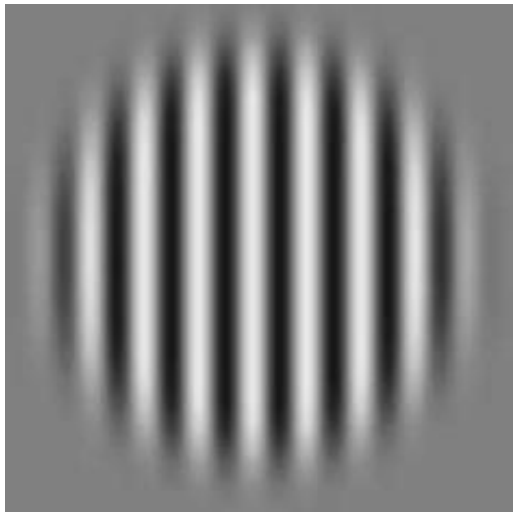
呈现图片

```
%example1_3 呈现一幅照片2秒钟。
%呈现一幅照片2秒钟。
ScreenNumber=0;
Background=128;
PicDuration=2;%图片2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间
ImgMatrix=imread('mypictures\fish.png'); %读取fish.png图片
ImgSize=size(ImgMatrix);
PicRect=CenterRect([0 0 ImgSize(2) ImgSize(1)],windowRect);%%图片呈现于屏幕中央,图片的位置矩阵
%%精确的写法,精确到帧
for ii=1:round(PicDuration/framedur)
    PicTexture=Screen('MakeTexture',WindowPtr, ImgMatrix);%把矩阵变为纹理
    Screen('DrawTexture',WindowPtr,PicTexture,[],PicRect);%画纹理
    Screen(WindowPtr,'Flip');%呈现图片（切换缓冲区）
end
Screen('CloseAll');%关闭窗口
```

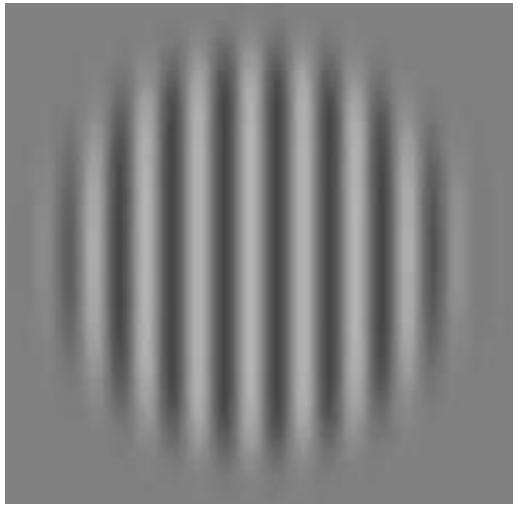
%example1_4 随机呈现一一系列图片，每幅图片呈现2秒钟。

```
%随机呈现一一系列图片，每幅图片呈现2秒钟。
ScreenNumber=0;
Background=128;
PicDuration=2;%图片2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间


for kk=1:10
    %%%%从1-18号图片中随机抽取一张呈现
    ImgIndex=randsample(1:18,1);
    ImgName=[num2str(ImgIndex) '.bmp'];
    ImgMatrix=imread(['mypictures\' ImgName]);
    ImgSize=size(ImgMatrix);
    PicRect=CenterRect([0 0 ImgSize(2) ImgSize(1)],windowRect);%%图片呈现于屏幕中央,图片的位置矩阵

    %%呈现图片
    for ii=1:round(PicDuration/framedur)
        PicTexture=Screen('MakeTexture',WindowPtr, ImgMatrix);%把矩阵变为纹理
        Screen('DrawTexture',WindowPtr,PicTexture,[],PicRect);%画纹理
        Screen(WindowPtr,'Flip');%呈现图片（切换纹理）
    end

end

Screen('CloseAll');%关闭窗口
```

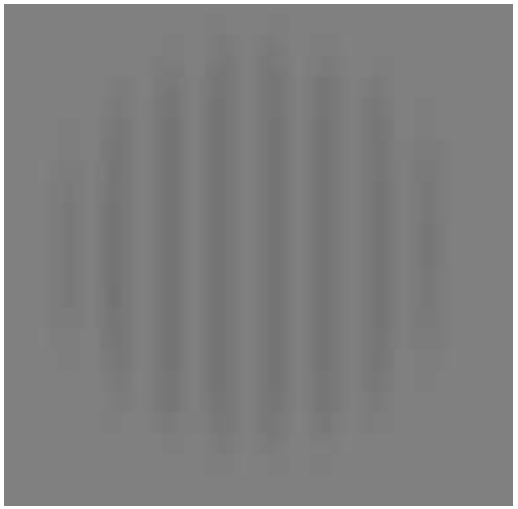
呈现光栅



高

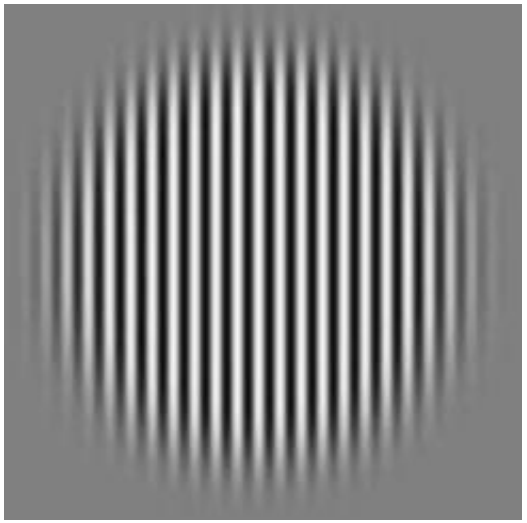


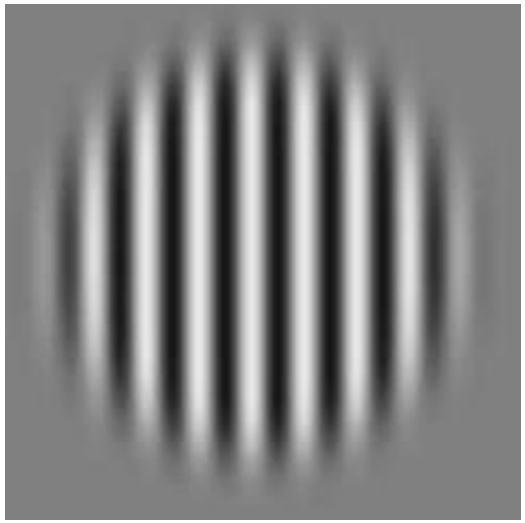
中

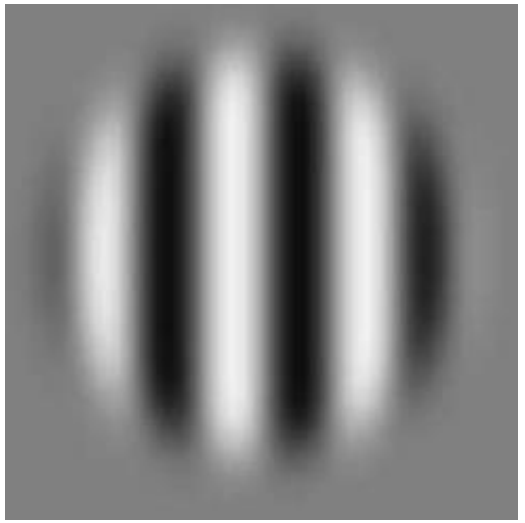


低

$$\text{Contrast（对比度）} = \frac{L_{最大} - L_{最小}}{L_{最大} + L_{最小}}$$





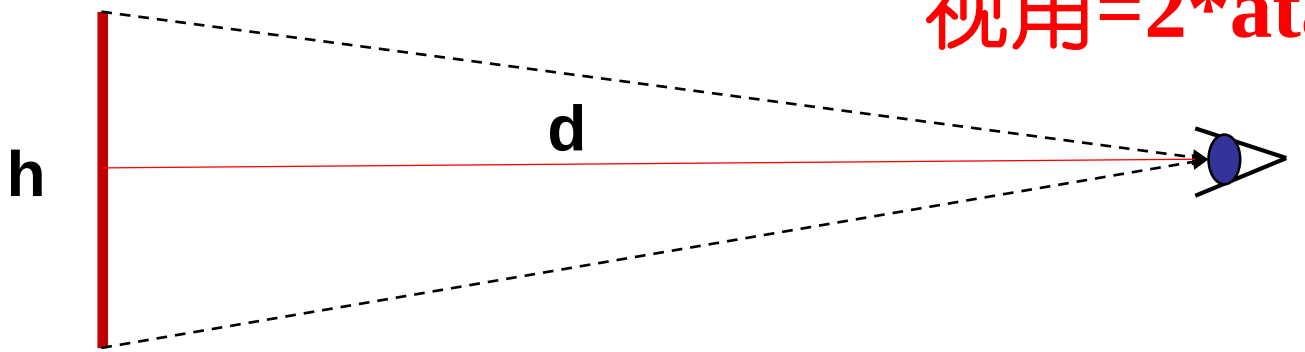


高　　中　　低

Spatial Frequency

（空间频率, cycles/degree）

视角=2*atan((h/2)/d)



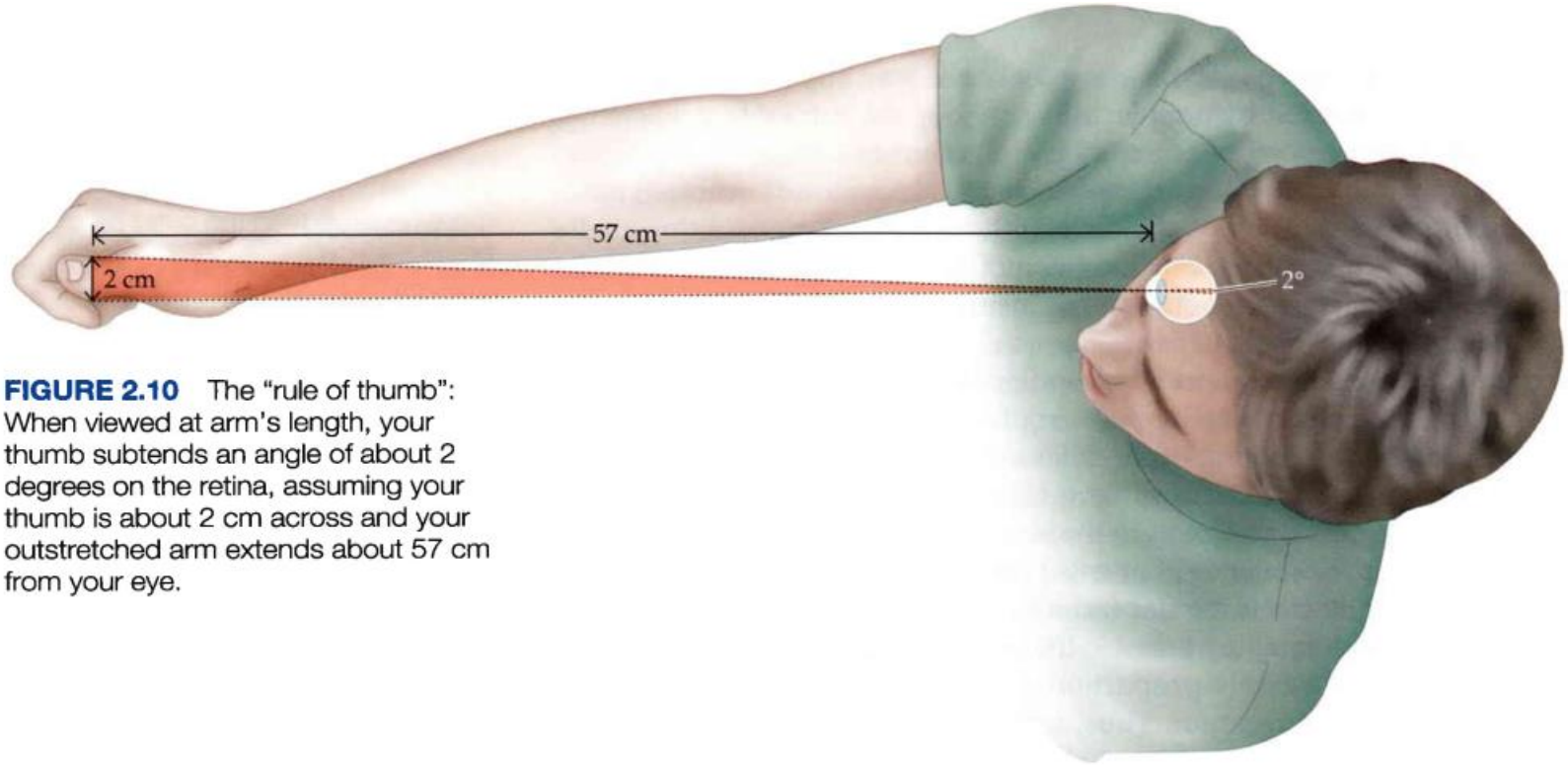
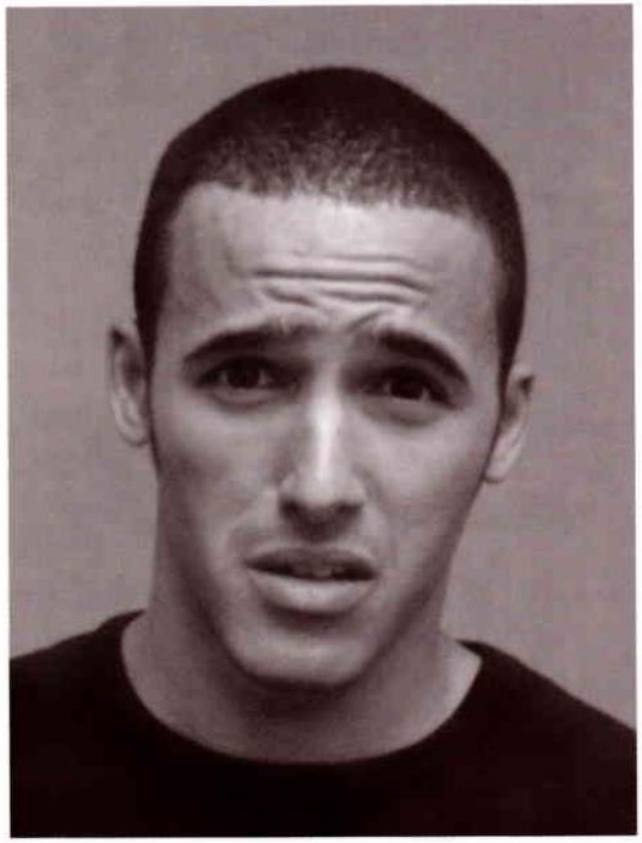


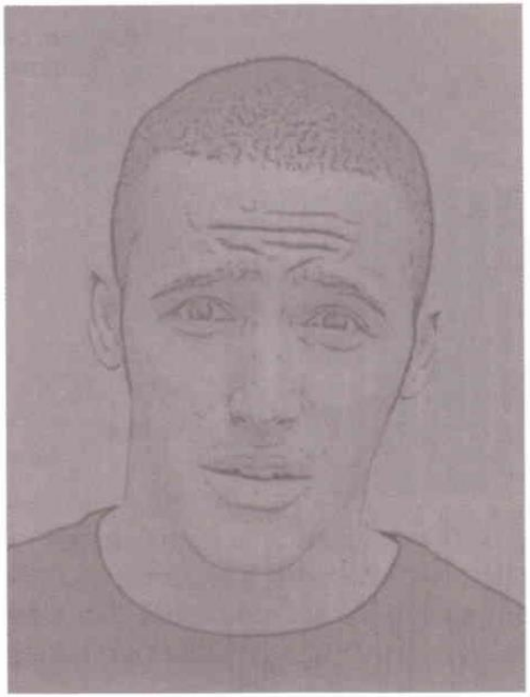
FIGURE 2.10 The "rule of thumb": When viewed at arm's length, your thumb subtends an angle of about 2 degrees on the retina, assuming your thumb is about 2 cm across and your outstretched arm extends about 57 cm from your eye.

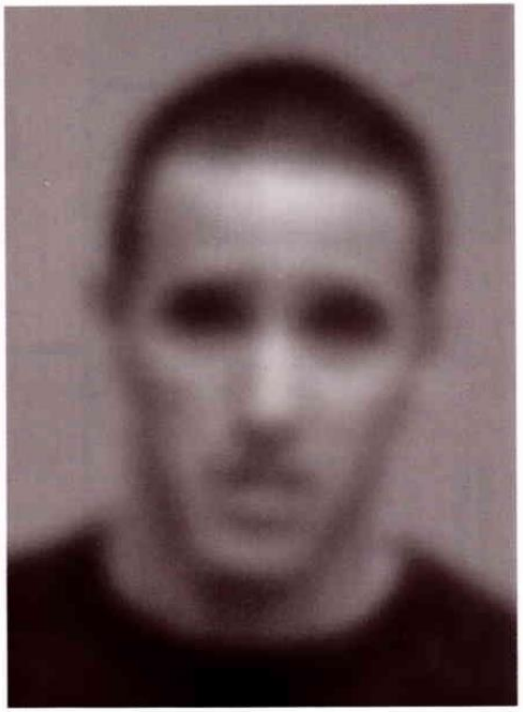
- 为什么用光栅作为实验刺激：



高通filter（滤波器）

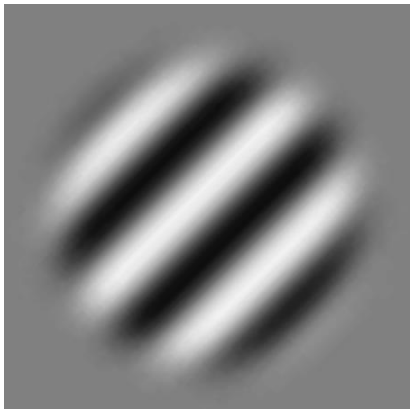
低通filter（滤波器）

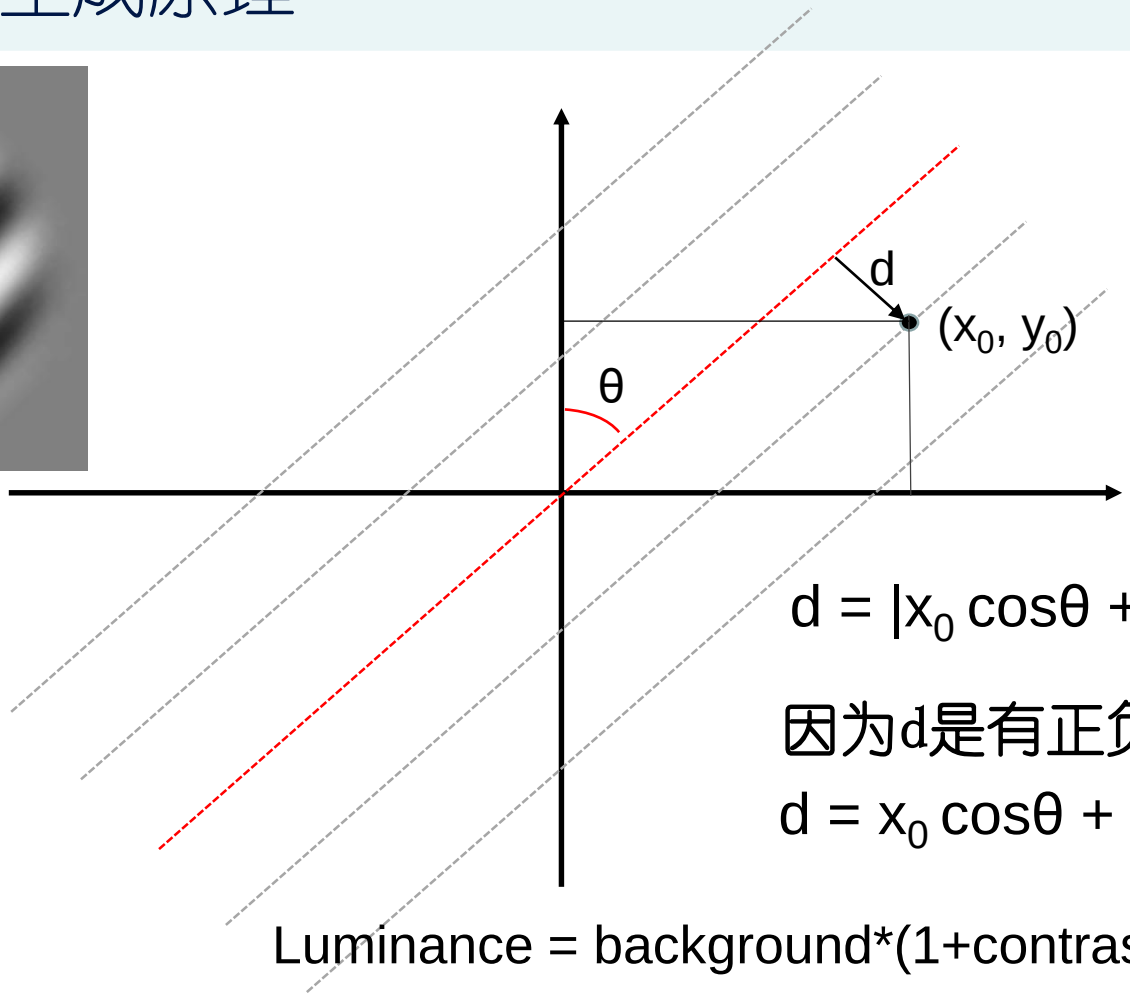




所有的图像都是由不同空间频率的正弦光栅叠加而成。

光栅的生成原理





$$d = |x_0 \cos\theta + y_0 \sin\theta|$$

因为d是有正负的，所以

$$d = x_0 \cos\theta + y_0 \sin\theta$$

Luminance = background*(1+contrast*sin(2*pi*sf*d))

```
sf=1/gratingperiod;   %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);
matrixofgrating=Background*(1+contrastofgrating*sin(a1*x+b1*y));    %生成光栅矩阵
```

%example1_5 画一个方形光栅，呈现2秒钟。

```
%画一个方形光栅
%By Cai Yongchun, 2020/10/7
ScreenNumber=0;
Background=128;
GratingDuration=5;%光栅呈现2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间

%定义光栅的参数
contrastofgrating=0.9; %对比度
sizeofgrating=200; %光栅大小
GratingRect=CenterRect([0 0 sizeofgrating sizeofgrating],windowRect);%%光栅呈现于屏幕中央,光栅的位置矩阵
[x,y]=meshgrid(-sizeofgrating/2:sizeofgrating/2,-sizeofgrating/2:sizeofgrating/2);
angleofgrating=60;  %光栅的倾斜角度，单位度
gratingperiod=20;  %空间周期
sf=1/gratingperiod;  %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);
matrixofgrating=Background*(1+contrastofgrating*sin(a1*x+b1*y));    %生成光栅矩阵
%surf(x,y,matrixofgrating)

GratingTexture=Screen('MakeTexture',WindowPtr, matrixofgrating);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(GratingDuration/framedur)
      Screen('DrawTexture',WindowPtr,GratingTexture,[],GratingRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现光栅
end
Screen('CloseAll');%关闭窗口
```

%example1_6 画一个圆形光栅，呈现2秒钟。

```
%画一个圆形光栅
ScreenNumber=0;
Background=128;
GratingDuration=2;%光栅呈现2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间

%定义光栅的参数
contrastofgrating=0.3; %对比度
sizeofgrating=200; %光栅大小
GratingRect=CenterRect([0 0 sizeofgrating sizeofgrating],windowRect);%%光栅呈现于屏幕中央,光栅的位置矩阵
[x,y]=meshgrid(-sizeofgrating/2:sizeofgrating/2,-sizeofgrating/2:sizeofgrating/2);
angleofgrating=30;  %光栅的倾斜角度，单位度
gratingperiod=100;  %空间周期
sf=1/gratingperiod;  %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);
maskradius=sizeofgrating/2;
Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask
matrixofgrating=Background*(1+0.8*sin(a1*x+b1*y).*Circlemask);    %生成光栅矩阵

surf(x,y,matrixofgrating)

GratingTexture=Screen('MakeTexture',WindowPtr, matrixofgrating);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(GratingDuration/framedur)
      Screen('DrawTexture',WindowPtr,GratingTexture,[],GratingRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现光栅
end
Screen('CloseAll');%关闭窗口
```

%example1_7 画一个Gabor光栅，呈现2秒钟。

```
%画一个gabor光栅
ScreenNumber=0;
Background=128;
GratingDuration=2;%光栅呈现2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间

%定义光栅的参数
contrastofgrating=0.3; %对比度
sizeofgrating=300; %光栅大小
GratingRect=CenterRect([0 0 sizeofgrating sizeofgrating],windowRect);%%光栅呈现于屏幕中央,光栅的位置矩阵
[x,y]=meshgrid(-sizeofgrating/2:sizeofgrating/2,-sizeofgrating/2:sizeofgrating/2);
angleofgrating=0;  %光栅的倾斜角度，单位度
gratingperiod=20;  %空间周期
sf=1/gratingperiod;  %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);
maskradius=sizeofgrating/2;
%Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask

sd=30;
Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维钟形mask
matrixofgrating=round(Background*(1+contrastofgrating*sin(a1*x+b1*y).*Circlemask));    %生成光栅矩阵

surf(x,y,matrixofgrating)

GratingTexture=Screen('MakeTexture',WindowPtr, matrixofgrating);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(GratingDuration/framedur)
      Screen('DrawTexture',WindowPtr,GratingTexture,[],GratingRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现光栅
end
Screen('CloseAll');%关闭窗口
```

%example1_8 画一个棋盘格，呈现2秒钟。

```
%画一个棋盘格
ScreenNumber=0;
Background=128;
GratingDuration=2;%光栅呈现2秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间
%定义光栅的参数
contrastofgrating=0.45; %对比度
sizeofgrating=300; %光栅大小
GratingRect=CenterRect([0 0 sizeofgrating sizeofgrating],windowRect);%%光栅呈现于屏幕中央,光栅的位置矩阵
[x,y]=meshgrid(-sizeofgrating/2:sizeofgrating/2,-sizeofgrating/2:sizeofgrating/2);
angleofgrating=0;  %光栅的倾斜角度，单位度
gratingperiod=40;  %空间周期
sf=1/gratingperiod;  %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);

a2=2*pi*sf*cos((angleofgrating+90)*pi/180);
b2=2*pi*sf*sin((angleofgrating+90)*pi/180);

maskradius=sizeofgrating/2;

sd=30;
Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维高斯mask
% Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask

%matrixofgrating=round(Background*(1+contrastofgrating*(sin(a1*x+b1*y)+sin(a2*x+b2*y)).*Circlemask));   %灰度棋盘格
matrixofgrating=round(Background*(1+contrastofgrating*sign(sin(a1*x+b1*y)+sin(a2*x+b2*y)).*Circlemask));   %黑白棋盘格
%surf(x,y,matrixofgrating)

GratingTexture=Screen('MakeTexture',WindowPtr, matrixofgrating);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(GratingDuration/framedur)
      Screen('DrawTexture',WindowPtr,GratingTexture,[],GratingRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现光栅
end
Screen('CloseAll');%关闭窗口
```

%example1_9 画一个风车图，呈现5秒钟。

```
%画一个风车图
ScreenNumber=0;
Background=128;
StimDuration=5;%呈现5秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间
%刺激的参数
StimContrast=0.45; %对比度
StimSize=300;%刺激大小
StimRect=CenterRect([0 0 StimSize StimSize],windowRect);%%刺激呈现于屏幕中央,刺激的位置矩阵
[x,y]=meshgrid(-StimSize/2:StimSize/2,-StimSize/2:StimSize/2);

maskradius=StimSize/2;

sd=50;
Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维高斯mask
Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask

cycles=6;
StimMatrix=Background*(1+StimContrast*sign(cos(cycles*atan(y./x))).*Circlemask); %风车的矩阵
surf(x,y,StimMatrix)

StimTexture=Screen('MakeTexture',WindowPtr, StimMatrix);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(StimDuration/framedur)
      Screen('DrawTexture',WindowPtr,StimTexture,[],StimRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现刺激
end
Screen('CloseAll');%关闭窗口
```

%example1_10 画一个随机点图，呈现5秒钟。

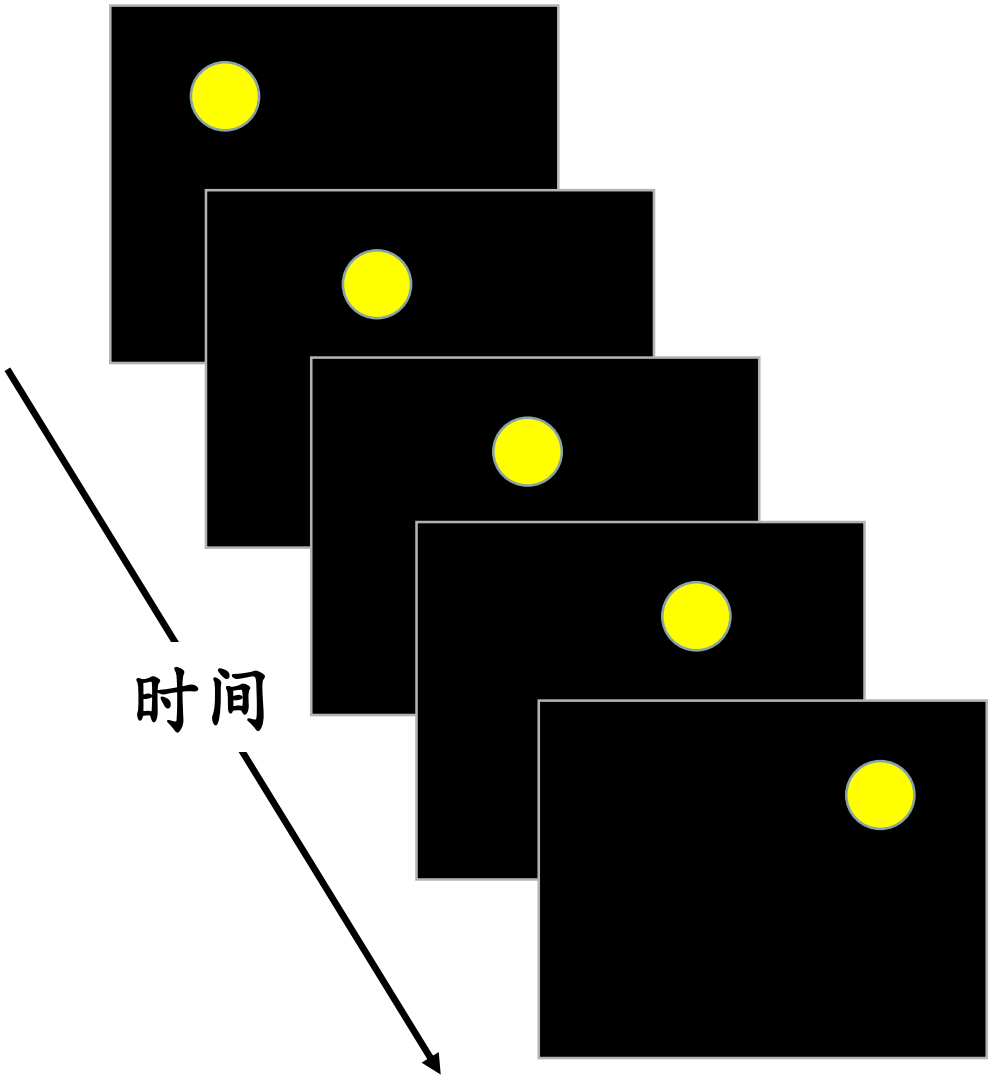
```
ScreenNumber=0;
Background=128;
StimDuration=5;%呈现5秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间
%刺激的参数
StimContrast=0.45; %对比度
StimSize=300;%刺激大小

NoisePix=6;%每个噪音点的像素数;
NoiseNum=round(StimSize/NoisePix);%水平或竖直方向上的噪音点个数
StimuSize=NoisePix*NoiseNum;%重新计算刺激大小

StimRect=CenterRect([0 0 StimSize StimSize],windowRect);%%刺激呈现于屏幕中央,刺激的位置矩阵
[x,y]=meshgrid(round(-StimSize/2):round(StimSize/2)-1,round(-StimSize/2):round(StimSize/2)-1);

maskradius=StimSize/2;

sd=50;
%Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维高斯mask
Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask

NoiseMatrix=rand(NoiseNum,NoiseNum)*2-1;
temp0=ones(NoisePix);
StimMatrix0=kron(NoiseMatrix,temp0);%生成像素噪音点，值范围-1~1

StimMatrix=Background*(1+StimContrast*StimMatrix0.*Circlemask); %刺激矩阵

surf(x,y,StimMatrix)

StimTexture=Screen('MakeTexture',WindowPtr, StimMatrix);%把矩阵变为纹理

%%呈现光栅
for ii=1:round(StimDuration/framedur)
      Screen('DrawTexture',WindowPtr,StimTexture,[],StimRect);%画纹理
      Screen(WindowPtr,'Flip');%呈现刺激
end
Screen('CloseAll');%关闭窗口
```

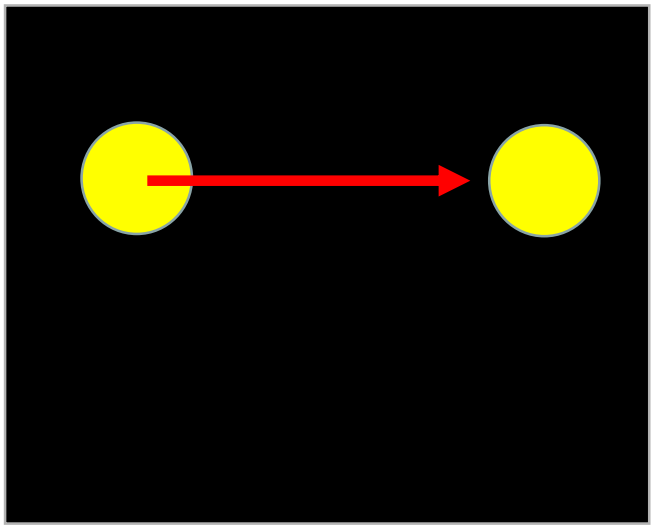
目 录

Outline

- 心理学编程需求及PTB
- 刺激的表达及显示原理
- 呈现静态图
- 呈现动态图

- 不同的静态图按一定的时间间隔呈现即形成动态刺激





%example1_11 画一个水平移动的圆形，按q键退出

```
% 一个圆在屏幕上水平运动，按q键退出
%By Cai Yongchun, 2020/10/7
Background=0;
StimColour=255;
StimSize=100;
DriftRange=[-400 400];%运动的范围，相对屏幕中央点
PixelsPerFrame=4;%每一帧运动的象素数
horizontaloffset=DriftRange(1):PixelsPerFrame:DriftRange(2);%可能的水平位置
%horizontaloffset=[DriftRange(1):PixelsPerFrame:DriftRange(2) DriftRange(2):-PixelsPerFrame:DriftRange(1)];%可能的水平位置
[WindowPtr,windowRect] =Screen('OpenWindow',0,Background);
TargetRect0=CenterRect([0 0 StimSize StimSize],windowRect);

quitekey=KbName('q');%q键退出
keycode=zeros(1,256);
n=0;
while ~keycode(quitekey)
    n=mod(n,length(horizontaloffset))+1;%水平位置编号
    TargetRect=TargetRect0+[horizontaloffset(n) 0 horizontaloffset(n) 0]; %每一帧都要计算新的刺激位置
    Screen('FillOval',WindowPtr,StimColour,TargetRect);
    Screen('Flip',WindowPtr);
    [keyisdown, secs, keycode]=KbCheck;%读取键盘值
end
Screen('CloseAll');
```

%example1_12 画一个运动的光栅，呈现5秒

```
%画一个运动的gabor光栅
%By Cai Yongchun, 2020/10/8
ScreenNumber=0;
Background=128;
GratingDuration=5;%光栅呈现5秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间

%定义光栅的参数
contrastofgrating=0.8; %对比度
sizeofgrating=300; %光栅大小
angleofgrating=0;  %光栅的倾斜角度，单位度
gratingperiod=20;  %空间周期
DriftSpeed=10; %光栅运动速度，单位像素/秒
GratingRect=CenterRect([0 0 sizeofgrating sizeofgrating],windowRect);%%光栅呈现于屏幕中央,光栅的位置矩阵
[x,y]=meshgrid(-sizeofgrating/2:sizeofgrating/2,-sizeofgrating/2:sizeofgrating/2);

sf=1/gratingperiod;  %转换成空间频率
a1=2*pi*sf*cos(angleofgrating*pi/180);
b1=2*pi*sf*sin(angleofgrating*pi/180);
maskradius=sizeofgrating/2;

sd=30;
Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维钟形mask
%Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask

%%呈现光栅
for ii=1:round(GratingDuration/framedur)
    matrixofgrating=round(Background*(1+contrastofgrating*sin(a1*x+b1*y-DriftSpeed*ii/frame_rate).*Circlemask));    %生成光栅矩阵
    GratingTexture=Screen('MakeTexture',WindowPtr, matrixofgrating);%把矩阵变为纹理
    Screen('DrawTexture',WindowPtr,GratingTexture,[],GratingRect);%画纹理
    Screen(WindowPtr,'Flip');%呈现光栅
end
Screen('CloseAll');%关闭窗口
```

%example1_13 画一个转动的风车，按q键退出

```
%画一个转动的风车
%By Cai Yongchun, 2020/10/8
ScreenNumber=0;
Background=128;
StimDuration=5;%呈现5秒钟
[WindowPtr,windowRect]=Screen('OpenWindow',ScreenNumber,Background);
frame_rate=Screen('FrameRate',WindowPtr);%获取刷新率
framedur=1/frame_rate;%每一帧呈现的时间

%刺激的参数
StimContrast=0.45; %对比度
StimSize=300;%刺激大小
StimRect=CenterRect([0 0 StimSize StimSize],windowRect);%%刺激呈现于屏幕中央,刺激的位置矩阵
[x,y]=meshgrid(-StimSize/2:StimSize/2,-StimSize/2:StimSize/2);
maskradius=StimSize/2;
sd=50;
Circlemask=exp(-(x.^2+y.^2)/(2*sd^2));%生成三维高斯mask
Circlemask=(x.^2+y.^2 <= maskradius^2);%生成圆形mask
cycles=6;
RotationSpeed=180;%转动速度，单位度/秒

%%呈现光栅
for ii=1:round(StimDuration/framedur)
    StimMatrix=Background*(1+StimContrast*sign(cos(cycles*atan(y./x)-(RotationSpeed*pi/180)*ii/frame_rate)).*Circlemask); %风车的矩阵
    StimTexture=Screen('MakeTexture',WindowPtr, StimMatrix);%把矩阵变为纹理
    Screen('DrawTexture',WindowPtr,StimTexture,[],StimRect);%画纹理
    Screen(WindowPtr,'Flip');%呈现刺激
end
Screen('CloseAll');%关闭窗口
```

谢谢各位！