Supplementary Materials and Methods

MATLAB script used for the conversion of RBG into HSL colour values. Note that this requires actual reflectance values for the six grayscale squares in the Xrite color checker; you will need to replace the rfl values with those of your own Xrite. This function reads a csv file with columns: directory, photo, patch, R, G, B, total reflectance. It creates a new output file with the results in the same number of rows as the input file.

This file contains several functions, where CalibrateRGBandSave calls the others, so everything in this document should be placed in an .m file, call it CalibrateRGBandSave.m When run it will ask for the input .csv file name.

```
function CalibrateRGBandSave;
%get fitting functions for each RGB value in each standard photograph
%assumes RGBs for grayscale standards in each standard photo in
RGBstandards.txt
rfl=zeros(6,1);
rfl(1)=0.0310; rfl(2)=0.0910; rfl(3)=0.1950;
rfl(4)=0.3720; rfl(5)=0.6090; rfl(6)=0.9480; %actual values for Xrite
grayscale (from Jair)

% rfl=log(rfl); %

%VARIABLES
*********************************************************************
% rfl(6)      Actual log reflectances of each grayscale square in
ColorChecker
%************************************************************************
*******

[fname,path]=uigetfile('*.csv','Select a csv photo data file');
fn=[path fname];
fid=fopen(fn);
rd=textscan(fid,'%s %s %s %s %s %s %s %s',1,'delimiter',',');
heads=[rd{1:end}];
rd=textscan(fid,'%s %s %s %f %f %f %f %s','delimiter',',');
fclose(fid);
heads=[heads 'Chroma' 'HueAngle'];

pdir=rd{1}; photo=rd{2}; patch=rd{3}; Red=rd{4}; Grn=rd{5}; Blu=rd{6};
Brt=rd{7};
Area=rd{8};
n=length(Red); clear fid rd;
%set up output file
k=strfind(fn,'.csv'); if isempty(k) k=0; end; k=k-1;
oname=[fn(1:k) 'Calibrated.csv'];
ofid=fopen(oname,'wt');
k=length(heads); fprintf(ofid,'%s',heads{1});
for i=2:k fprintf(ofid,',%s',heads{i}); end;
fprintf(ofid,'\n');

%identify each photo in order to extact standards and calibrate RGB
phlist=unique(photo); np=length(phlist);
phid=zeros(np,1);
```

```
for i=1:n
  str=photo{i};
  phid(i)=find(strcmp(phlist,str));
end; clear i str;
%do each photo
for p=1:np  %photo number p
  pdr=pdir(phid==p); pho=photo(phid==p); pch=patch(phid==p);
ars=Area(phid==p);
  R=Red(phid==p); G=Grn(phid==p); B=Blu(phid==p); BT=Brt(phid==p);
nd=length(BT);
  %1-24 are the standards  25:nd are the bird samples
  %1 is black 6 is white, corresponding to refl(1:6);
  rbd=R(25:nd); gbd=G(25:nd); bbd=B(25:nd); btbd=BT(25:nd); %bird
measurements
  pdbd=pdr(25:nd); phbd=pho(25:nd); ptchbd=pch(25:nd); arbd=ars(25:nd);
%bird metadata
  rst=R(1:6); gst=G(1:6); bst=B(1:6); btst=BT(1:6);         %gray
standards
%  %plot raw data actual on x axis
%  figure;
%  plot(rfl,rst,'r',rfl,rst,'or'); hold on;
%  plot(rfl,gst,'g',rfl,gst,'og');
%  plot(rfl,bst,'b',rfl,bst,'ob');
%  plot(rfl,btst,'k--',rfl,btst,'ok'); hold off; %log shape, log(rfl) not
quite linear
%  xlabel('actual reflectance'); ylabel('R,G,B,BT'); hold on;

 %get function to correct RGB to white (R=G=B), first fit lines to all
points
  x=(0.03:0.005:1)';
  [xData,yData]=prepareCurveData(rfl,rst); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  fitted=fit(xData,yData,ft,opts); yr=feval(fitted,x);
  [xData,yData]=prepareCurveData(rfl,gst); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  fitted=fit(xData,yData,ft,opts); yg=feval(fitted,x);
  [xData,yData]=prepareCurveData(rfl,bst); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  fitted=fit(xData,yData,ft,opts); yb=feval(fitted,x);
  [xData,yData]=prepareCurveData(rfl,btst); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  fitted=fit(xData,yData,ft,opts); ybt=feval(fitted,x);
  clear xData Ydata opts fitted ft;
  % yr,yg,yb,ybt are the continuous best fits to the grayscale data
%     figure;
%     plot(rfl,rst,'or',rfl,gst,'og',rfl,bst,'ob',rfl,btst,'ok'); hold
on;
%     plot(x,yr,'r',x,yg,'g',x,yb,'b',x,ybt,'k'); xlabel('actual total
reflectance');
```

```
%     ylabel('R,G,B,Brt'); title('fitted data for irradiance
correction');
  %excange x y and get fits from RGB to actual reflectance
  [xData,yData]=prepareCurveData(yr,x); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  eqR=fit(xData,yData,ft,opts);
  [xData,yData]=prepareCurveData(yg,x); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  eqG=fit(xData,yData,ft,opts);
  [xData,yData]=prepareCurveData(yb,x); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  eqB=fit(xData,yData,ft,opts);
  [xData,yData]=prepareCurveData(ybt,x); ft=fittype('exp2');
  opts=fitoptions('Method','NonlinearLeastSquares');
  opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
  eqBt=fit(xData,yData,ft,opts);
%     figure; plot(yr,x,'r',yg,x,'g',yb,x,'b',ybt,x,'k:'); hold on;
%     xlabel('R,G,B,Bt'); ylabel('reflectance+light');
%     title('Fitted data to convert to light+reflectance');
%     yR=feval(eqR,rst); plot(rst,yR,'or'); yG=feval(eqG,rst);
plot(rst,yG,'og');
%     yB=feval(eqB,rst); plot(rst,yB,'ob'); yBt=feval(eqBt,rst);
plot(rst,yBt,'ok');
  %convert rgb for birds to reflectance+light
  % rbd gbd bbd btbd bird measurements,  heads has original headings--use
again
  % pdbd directory, phbd photo, ptchbd patch name, arbd area on bird
%bird metadata
  r=feval(eqR,rbd); g=feval(eqG,gbd); b=feval(eqB,bbd);
bt=feval(eqBt,btbd);
  nb=length(r); %number of bird samples
  %use rst,gst,bst,btst(4) to get white balance coefficients
  wh=[rst(4) gst(4) bst(4) btst(4)]; whc=wh/wh(4); %like the von Kries!
  cal=zeros(nb,4); %corrected for both reflectance and white balance
(illumination)
  for s=1:nb
    bd=[r(s) g(s) b(s) bt(s)]./whc;
    cal(s,:)=bd;
  end;
  for s=1:nb
    fprintf(ofid,'%s,%s,%s',pdbd{s},phbd{s},ptchbd{s});
    for k=1:4 fprintf(ofid,',%6.4f',cal(s,k)); end;
    fprintf(ofid,',%s',arbd{s});
    rc=cal(s,1); gc=cal(s,2); bc=cal(s,3); t=rc+gc+bc;
    rc=rc/t; gc=gc/t; bc=bc/t;
    [hue,chr]=TriToHSV(rc,gc,bc);
    fprintf(ofid,',%6.4f,%6.2f\n',chr,hue);
  end;
  fprintf(1,'Finished photo %s\n',phbd{1});
end; %photo number p
fclose(ofid);
```

```
fprintf(1,'Finished file %s \n   in directory %s\n',fname,path);
end

function [hue,chr]=TriToHSV(R,G,B)
% [hue,chr]=TriToHSV(R,G,B);
%get HSV from calibrated RGB in Triangle
%INPUT   R,G,B  calibrated R G and B values as column vectors
%OUTPUT  hue    hue angle R at -120, G at 0, B at +120 degrees
%        chr    chroma (maximum possible 1.0)
%
[xa,ya]=ToTriangle([1 0 0]); %vertex for maximum chroma 0.5774
[xw,yw]=ToTriangle([1/3 1/3 1/3]); %  0.5000    0.2887
[xc,yc]=ToTriangle([R G B]);
xa=xa-xw; ya=ya-yw;
xc=xc-xw; yc=yc-yw; %make gray pont at 0,0
[h,c]=cart2pol(ya,xa); %max chroma value 0.5774
[hue,chr]=cart2pol(yc,xc);
hue=(180/pi)*hue;  chr=chr/c; %red at -120, green at 0, blue at +120
degrees
end

function [tx,ty]=ToTriangle(xyz)
% [tx,ty]=ToTriangle(xyz)
%    converts data matrix xyz=[x y z] of x,y,z coordinates
%    to triangular coordinates tx,ty (both column vectors)
%    with sides=1, height sqrt(3)/2
% Will temporarily make xyz rows sum to 1
% Coordinates of triangle are trix=[0 1 0.5 0]; triy=[0 0 sqrt(3)/2 0];
% Same as plottri but only saves triangular coordinates
[N,C]=size(xyz); ht=sqrt(3)/2;
su=sum(xyz')'; for i=1:N xyz(i,:)=xyz(i,:)/su(i); end; %row sums=1
sx=(sqrt(3)/2)*xyz; % sx=xyz to make height=1 instead of sides
for i=1:N tx(i)=(sx(i,2)+2*sx(i,3))/sqrt(3); ty(i)=sx(i,2); end;
tx=tx'; ty=ty'; %convert to column vectors
end
```

function DigitizeFrogsIntactXriteCR2files

```
%  DigitizeFrogsIntactXriteCR2files;
%digitize frog pictures, intact XRITE standard
%this version for CR2 files
% epm509@uowmail.edu.au   srk649@uowmail.edu.au
clear;

head='fmeanR,fmeanG,fmeanR,fmeanTot,fsdR,fsdG,fsdB,fsdTot';
head=[head ',hmeanR,hmeanG,hmeanR,hmeanTot,hsdR,hsdG,hsdB,hsdTot'];
head=[head ',tmeanR,tmeanG,tmeanR,tmeanTot,tsdR,tsdG,tsdB,tsdTot'];
head=[head ',vmeanR,vmeanG,vmeanR,vmeanTot,vsdR,vsdG,vsdB,vsdTot,Photo'];

[dirs,nds]=GetDIRList('./');
[s,v]=listdlg('ListString',dirs,'SelectionMode','single',...
    'PromptString','Select directory with photo files','Name','Select
Directory',...
    'ListSize',[200 (20*nds)]);
 dname=dirs{s}; dr=['./' dname '/'];

 oname=['YellowPatchData' dname '.csv'];
 fid=fopen(oname);
 if fid<0 new=1; else new=0; fclose(fid); end;
 if new==1
   fid=fopen(oname,'wt');  %create new file for output data
   fprintf(fid,'Calibrated RGB and total reflectance estimates for frog,
head(h), torso(t), vent(v) in directory "%s"\n',dr);
   fprintf(fid,'%s\n',head);
 else
   fid=fopen(oname,'at');  %append to existing file if present
 end;

 [files,nf]=GetFileList(dr,'.CR2');
 if nf <1
   fprintf(1,'No .CR2 files found in directory %s\n',dname);
   fclose('all');
   stop;
 end;

 v=1;
 while v>0
  [s,v]=listdlg('ListString',files,'SelectionMode','single',...
    'PromptString',{'Select photo file to use','Cancel to end pgm'},...
    'Name','Select Photo');
  close all;
  if v>0
    fname=files{s}; fprintf(1,'doing %s\n',fname);
    fn=[dr fname];
    img=imread(fn);
    [rows,cols,d]=size(img); if rows>cols img=imrotate(img,270); end;
    [rows,cols,d]=size(img);
    img=flipud(img); %CR2 files flipped up and down
    imshow(img,'initialmagnification','fit');

    ch=questdlg('Is grayscale on right edge of Xrite','Image
mirrored?','Yes','No','Yes'); %last one default
    switch ch  %some cr2 files dark, rescale for outlining only
     case 'Yes'
        dark=0;
     case 'No'
        img=fliplr(img);
    end;
    imshow(img,'initialmagnification','fit');

    ch=questdlg('Is image upside-down','Image
mirrored?','Yes','No','No'); %last one default
    switch ch  %some cr2 files dark, rescale for outlining only
     case 'Yes'
        img=flipud(img);
     case 'No'
        dark=0;
    end;
    imshow(img,'initialmagnification','fit');

    dark=0;
    ch=questdlg('Is this image dark?','Image Dark?','Yes','No','No');
%last one default
    switch ch  %some cr2 files dark, rescale for outlining only
     case 'Yes'
        simg=double(img);
```

```
            simg(img>128)=128; simg=2.2*simg; simg=uint8(simg);
            dark=1;
         case 'No'
            simg=img;
            dark=0;
        end;

        figure(1); set(gcf,'Position',[816 270 1102 860]);
        imshow(simg,'initialmagnification','fit');
        drawnow;

        msg='CLICK carefully ON 4 WHITE CORNER MARKS, adjust, then double-
    click on corner';
        hold on;  drawnow;
        title(msg,'FontSize',16,'FontWeight','bold');
        drawnow;
        [mask,mx,my]=roipoly(simg); hold on; plot(mx,my,'w');
        x1=mx(1); x2=mx(2); y1=my(1); y2=my(2);
        dst=sqrt((x1-x2)^2 + (y1-y2)^2);
        scale=dst/60;  %pixels/mm, 60mm between corners at narrower dimension
        clear x1 x2 y1 y2 dst;

        msg='CLICK and drag to CROP around frog, then double-click on last
    corner';
        title(msg,'FontSize',16,'FontWeight','bold');
        drawnow;
        frogimg=imcrop(img);
      %   figure; imshow(frogimg);
      %   msg='CLICK around edge of frog''s body, then double-click on first
    point';
      %   title(msg,'FontSize',16,'FontWeight','bold');
      %   [frmask,xx,yy]=roipoly(frogimg); hold on; plot(xx,yy,'w'); clear xx
    yy;
      %   title('Body pattern outlined','FontSize',16,'FontWeight','bold');
    drawnow;
      %   figure(1);
        title('Getting standard RGBs & frog patches, takes a few seconds',...
            'FontSize',16,'FontWeight','bold'); drawnow;
        %find angle to make mx,my aligned 0 and 90 degrees
        x=mx(2)-mx(1); y=my(2)-my(1); [rot,R] = cart2pol(x,y);
        %temporary rotation of outline
        [pcx,pcy]=polycenter(mx,my); tmx=mx-pcx; tmy=my-pcy; %center on 0,0
        [ang,dst]=cart2pol(tmx,tmy); [tx,ty]=pol2cart((ang-rot),dst);
        [X,Y]=meshgrid(1:4,1:6);
        xrange=abs(max(tx)-min(tx)); yrange=abs(max(ty)-min(ty));
        xin=xrange/4; yin=yrange/6; xs=min(tx)+xrange/8;
    ys=min(ty)+yrange/12;
        x=(xs-xin)+X*xin; y=(ys-yin)+Y*yin;
        %rotate back
        [ang,dst]=cart2pol(x,y); [px,py]=pol2cart((ang+rot),dst);
        px=px+pcx; py=py+pcy;
        clear x y rot R pcx pcy tmx tmy ang dst tx ty X Y xrange yrange xs
    ys;
        cx=px; cy=py; % plot(cx,cy,'ok'); %cx,cy contain square centres
        xr=xin/3.4; yr=yin/3.4; px=zeros(5,1); py=px;
        masks=cell(24,1);
        for c=1:24
          x=cx(c); y=cy(c);
          px(1)=x-xr; py(1)=y-yr; px(2)=x+xr; py(2)=y-yr;
          px(3)=x+xr; py(3)=y+yr; px(4)=x-xr; py(4)=y+yr; px(5)=px(1);
    py(5)=py(1);
          plot(px,py,'g'); text(x,y,num2str(c));
          %save each mask and extract r,g,b,gry from each
          msk=roipoly(img,px,py); masks{c}=msk;
        end;
        clear x y px py msk  ;
        r=zeros(rows,cols); g=r; b=r;
        r=img(:,:,1); g=img(:,:,2); b=img(:,:,3); gry=rgb2gray(img);
        rgbg=zeros(24,4);
        for c=1:24
          msk=masks{c}; stn=num2str(c);
          mr=mean(r(msk==1)); mg=mean(g(msk==1)); mb=mean(b(msk==1));
          mgy=mean(gry(msk==1));
          rgbg(c,:)=[mr mg mb mgy];
        end;
        clear r g b c* d masks msk stn mr mg mb mgy mx my cols rows x* y* msg
    gry mask simg;
        title(['doing frog in '
    fname],'FontSize',16,'FontWeight','bold','interpreter','none');
        drawnow;
        %VARIABLES
    ************************************************************************
        %  fn      file name
        %  scale   %pixels/mm, derived from 50mm in scale
        %  img     original colour image
        %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
    standards
        %  frogimg frog colour image, cropped for greater efficiency

    %**************************************************************************
    *********
        %in order to threshold frogimage, need to rescale
        test=double(frogimg);
        if dark>0
           test(test>125)=125; test=uint8(4*test);  %dark images
        else
           test=uint8(2*test);                      %light images
        end;
        % imshow(test);  %this is only for thresholding, not for RGB values



    %**************************************************************************
    *******
        figure; imshow(test); hold on;
        r=test(:,:,1); g=test(:,:,2); b=test(:,:,3); [rw,cl]=size(r);
        h=text(50,50,'<CR> then draw around head'); drawnow; pause;
        delete(h); [maskhead,pc1,pr1]=roipoly(test); plot(pc1,pr1,'b--');
        testhead=uint8(zeros(rw,cl,3));
        rm=r; rm(maskhead==0)=255; gm=g; gm(maskhead==0)=255; bm=b;
    bm(maskhead==0)=255;
        testhead(:,:,1)=rm; testhead(:,:,2)=gm; testhead(:,:,3)=bm;

        h2=text(50,50,'<CR> then draw around torso'); drawnow; pause
        delete(h2); [masktorso,pc2,pr2]=roipoly(test); plot(pc2,pr2,'b--');
        testtorso=uint8(zeros(rw,cl,3));
        rm=r; rm(masktorso==0)=255; gm=g; gm(masktorso==0)=255; bm=b;
    bm(masktorso==0)=255;
        testtorso(:,:,1)=rm; testtorso(:,:,2)=gm; testtorso(:,:,3)=bm;
        delete(h);

        h3=text(50,50,'<CR> then draw around vent'); drawnow; pause
        delete(h3); [maskvent,pc,pr]=roipoly(test);
        plot(pc1,pr1,'r--',pc2,pr2,'g--',pc,pr,'b--');
        testvent=uint8(zeros(rw,cl,3));
        rm=r; rm(maskvent==0)=255; gm=g; gm(maskvent==0)=255; bm=b;
    bm(maskvent==0)=255;
        testvent(:,:,1)=rm; testvent(:,:,2)=gm; testvent(:,:,3)=bm;
        clear mask* rm gm bm; delete(h);
        % testhead,testtorso,testvent contain 3 parts, test contains all
        h4=text(50,50,'<CR> to continue'); drawnow; pause;
        delete(h4);


    %*************************************************************************
    ********
        %do entire frog image
        r=test(:,:,1); g=test(:,:,2); b=test(:,:,3);
        df=imlincomb(0.5,r,0.5,g,-1,b); bw=im2bw(df,0.2); % figure(2);
    imshow(bw);
        cc=bwconncomp(bw);
        if dark>0
          crts=40;
        else
          crts=20;
        end;
        stats=regionprops(cc,'Area'); idx=find([stats.Area] > crts);
        bw2=ismember(labelmatrix(cc),idx);
        %  bdy=bw; bdy(frmask==0)=0;  %restrict to outlined parts
        if dark>0
          se=strel('disk',6);
        else
          se=strel('disk',2);     %erode to avoid boundaries
        end;
        bw2=imerode(bw2,se,12); % figure; imshow(bw2);
        yellow=bw2; %contains pixels to be measured (yellow to orange spots)
        % figure; imshow(yellow)
        fr=test(:,:,1); fg=test(:,:,2); fb=test(:,:,3);
        fr(yellow==1)=0; fg(yellow==1)=0; fb(yellow==1)=150;
        shw=test; shw(:,:,1)=fr; shw(:,:,2)=fg; shw(:,:,3)=fb;
       figure(2); set(gcf,'Position',[842 292 1068 834]);
    subplot(1,2,1); imshow(test); subplot(1,2,2); imshow(shw);
        title([fname ', blue=measured'],'interpreter','none',...
            'FontSize',16,'FontWeight','bold');
        clear x1 x2 y1 y2 slp ic r g b df bw bw2 stats fr fg fb tle tls idx
    cc shw;
        %VARIABLES
    **********************************************************************
        %  fn      file name
        %  scale   %pixels/mm, derived from 50mm in scale
        %  img     original colour image
        %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
    standards
```

```
    %  frogimg(rows,cols,3) frog colour image, cropped for greater
efficiency
    %  yellow(rows,cols)    mask: white for yellow blotches away from
edges

%***************************************************************************
********

    % testhead,testtorso,testvent contain 3 parts, test contains all

%***************************************************************************
********
    %do head image
    r=testhead(:,:,1); g=testhead(:,:,2); b=testhead(:,:,3);
    df=imlincomb(0.5,r,0.5,g,-1,b); bw=im2bw(df,0.2); % figure(2);
imshow(bw);
    cc=bwconncomp(bw);
    if dark>0
      crts=40;
    else
      crts=20;
    end;
    stats=regionprops(cc,'Area'); idx=find([stats.Area] > crts);
    bw2=ismember(labelmatrix(cc),idx);
    %  bdy=bw; bdy(frmask==0)=0;  %restrict to outlined parts
    if dark>0
      se=strel('disk',6);
    else
      se=strel('disk',2);     %erode to avoid boundaries
    end;
    bw2=imerode(bw2,se,12); % figure; imshow(bw2);
    yellowhead=bw2; %contains pixels to be measured (yellow to orange
spots)
    % figure; imshow(yellowhead)
    clear x1 x2 y1 y2 slp ic r g b df bw bw2 stats fr fg fb tle tls idx
cc shw;
    %VARIABLES
**********************************************************************
    %  fn      file name
    %  scale   %pixels/mm, derived from 50mm in scale
    %  img     original colour image
    %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
standards
    %  frogimg(rows,cols,3) frog colour image, cropped for greater
efficiency
    %  yellow(rows,cols)     mask: white for yellow blotches away from
edges
    %  yellowhead(rows,cols) same for head region

%***************************************************************************
********

    % testhead,testtorso,testvent contain 3 parts, test contains all

%***************************************************************************
********
    %do torso image
    r=testtorso(:,:,1); g=testtorso(:,:,2); b=testtorso(:,:,3);
    df=imlincomb(0.5,r,0.5,g,-1,b); bw=im2bw(df,0.2); % figure(2);
imshow(bw);
    cc=bwconncomp(bw);
    if dark>0
      crts=40;
    else
      crts=20;
    end;
    stats=regionprops(cc,'Area'); idx=find([stats.Area] > crts);
    bw2=ismember(labelmatrix(cc),idx);
    %  bdy=bw; bdy(frmask==0)=0;  %restrict to outlined parts
    if dark>0
      se=strel('disk',6);
    else
      se=strel('disk',2);     %erode to avoid boundaries
    end;
    bw2=imerode(bw2,se,12); % figure; imshow(bw2);
    yellowtorso=bw2; %contains pixels to be measured (yellow to orange
spots)
    % figure; imshow(yellowtorso)
    clear x1 x2 y1 y2 slp ic r g b df bw bw2 stats fr fg fb tle tls idx
cc shw;
    %VARIABLES
*********************************************************************
    %  fn      file name
    %  scale   %pixels/mm, derived from 50mm in scale
    %  img     original colour image
    %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
standards
    %  frogimg(rows,cols,3) frog colour image, cropped for greater
efficiency
    %  yellow(rows,cols)      mask: white for yellow blotches away from
edges
    %  yellowhead(rows,cols)  same for head region
    %  yellowtorso(rows,cols) same for torso region

%************************************************************************
********

    % testhead,testtorso,testvent contain 3 parts, test contains all

%************************************************************************
********
    %do vent image
    r=testvent(:,:,1); g=testvent(:,:,2); b=testvent(:,:,3);
    df=imlincomb(0.5,r,0.5,g,-1,b); bw=im2bw(df,0.2); % figure(2);
imshow(bw);
    cc=bwconncomp(bw);
    if dark>0
      crts=40;
    else
      crts=20;
    end;
    stats=regionprops(cc,'Area'); idx=find([stats.Area] > crts);
    bw2=ismember(labelmatrix(cc),idx);
    %  bdy=bw; bdy(frmask==0)=0;  %restrict to outlined parts
    if dark>0
      se=strel('disk',6);
    else
      se=strel('disk',2);      %erode to avoid boundaries
    end;
    bw2=imerode(bw2,se,12); % figure; imshow(bw2);
    yellowvent=bw2; %contains pixels to be measured (yellow to orange
spots)
    % figure; imshow(yellowtorso)
    clear x1 x2 y1 y2 slp ic r g b df bw bw2 stats fr fg fb tle tls idx
cc shw;
    %VARIABLES
*********************************************************************
    %  fn       file name
    %  scale    %pixels/mm, derived from 50mm in scale
    %  img      original colour image
    %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
standards
    %  frogimg(rows,cols,3) frog colour image, cropped for greater
efficiency
    %  yellow(rows,cols)      mask: white for yellow blotches away from
edges
    %  yellowhead(rows,cols)   same for head region
    %  yellowtorso(rows,cols) same for torso region
    %  yellowvent(rows,cols) same for torso region

%************************************************************************
********
%
%     figure;
%     subplot(2,2,1); imshow(yellow);
%     subplot(2,2,2); imshow(yellowhead);
%     subplot(2,2,3); imshow(yellowtorso);
%     subplot(2,2,4); imshow(yellowvent);
%

    % figure; imshow(yellow);
    frogimg=double(frogimg);
    fr=frogimg(:,:,1); fg=frogimg(:,:,2); fb=frogimg(:,:,3);
    frf=fr(yellow==1);  fgf=fg(yellow==1);  fbf=fb(yellow==1);
%entire frog
    frh=fr(yellowhead==1);  fgh=fg(yellowhead==1);
fbh=fb(yellowhead==1);  %head
    frt=fr(yellowtorso==1); fgt=fg(yellowtorso==1);
fbt=fb(yellowtorso==1); %torso
    frv=fr(yellowvent==1);  fgv=fg(yellowvent==1);
fbv=fb(yellowvent==1);  %vent

    clear yellow img;
    %VARIABLES
**********************************************************************
    %  fn       file name
    %  scale    %pixels/mm, derived from 50mm in scale
    %  rgbg(24,4) mean RGB and Gray values for the 24 colour & gray
standards
    %  frogimg(rows,cols,3) frog colour image, cropped for greater
efficiency
    %  frf,fgf,fbf  RGB values for all yellow stripe pixels in frog
    %  frh,fgh,fbh  RGB values for all yellow stripe pixels in frog's
head
```

```
    %  frt,fgt,fbt  RGB values for all yellow stripe pixels in frog's
torso
    %  frv,fgv,fbv  RGB values for all yellow stripe pixels in frog's
vent
    %    to view frogimg imshow(uint8(frogimg))

%***********************************************************************
*********
    rfl=zeros(6,1);  %gray standards
    rfl(1)=0.0310; rfl(2)=0.0910; rfl(3)=0.1950;
    rfl(4)=0.3720; rfl(5)=0.6090; rfl(6)=0.9480; %actual values for Xrite
grayscale
    rst=rgbg(19:24,1); gst=rgbg(19:24,2); bst=rgbg(19:24,3);
btst=rgbg(19:24,1);
  %   %plot raw data vs actual
  %   figure;
  %   plot(rfl,rst,'r',rfl,rst,'or'); hold on;
  %   plot(rfl,gst,'g',rfl,gst,'og');
  %   plot(rfl,bst,'b',rfl,bst,'ob');
  %   plot(rfl,btst,'k--',rfl,btst,'ok'); hold off; %log shape, log(rfl)
not quite linear
  %   xlabel('actual reflectance'); ylabel('R,G,B,BT');
    %get calibration
    %get function to correct RGB to white (R=G=B), first fit lines to all
points
    x=(0.03:0.005:1)';
    [xData,yData]=prepareCurveData(rfl,rst); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    fitted=fit(xData,yData,ft,opts); yr=feval(fitted,x);
    [xData,yData]=prepareCurveData(rfl,gst); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    fitted=fit(xData,yData,ft,opts); yg=feval(fitted,x);
    [xData,yData]=prepareCurveData(rfl,bst); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    fitted=fit(xData,yData,ft,opts); yb=feval(fitted,x);
    clear xData ydata opts fitted ft;
    % yr,yg,yb are the continuous best fits to the grayscale data
  %   figure;
  %   plot(rfl,rst,'or',rfl,gst,'og',rfl,bst,'ob'); hold on;
  %   plot(x,yr,'r',x,yg,'g',x,yb,'b'); xlabel('actual total
reflectance');
  %   ylabel('R,G,B,Brt'); title('fitted data for irradiance
correction');
    % excange x y and get fits from RGB to actual reflectance
    [xData,yData]=prepareCurveData(yr,x); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    eqR=fit(xData,yData,ft,opts);
    [xData,yData]=prepareCurveData(yg,x); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    eqG=fit(xData,yData,ft,opts);
    [xData,yData]=prepareCurveData(yb,x); ft=fittype('exp2');
    opts=fitoptions('Method','NonlinearLeastSquares');
    opts.StartPoint=[0.0329939089339003 0.0136159137399716 -
0.0983277644336392 0.00351296038818105];
    eqB=fit(xData,yData,ft,opts);
    clear xData ydata opts fitted ft;
  %     figure; plot(yr,x,'r',yg,x,'g',yb,x,'b'); hold on;
  %     xlabel('R,G,B,Bt'); ylabel('reflectance+light');
  %     title('Fitted data to convert to light+reflectance');
  %     yR=feval(eqR,rst); plot(rst,yR,'or'); yG=feval(eqG,rst);
plot(rst,yG,'og');
  %     yB=feval(eqB,rst); plot(rst,yB,'ob');

    %convert to estimated reflectance in each channel and luminance
    %  frf,fgf,fbf  RGB values for all yellow stripe pixels in frog
    %  frh,fgh,fbh  RGB values for all yellow stripe pixels in frog's
head
    %  frt,fgt,fbt  RGB values for all yellow stripe pixels in frog's
torso
    %  frv,fgv,fbv  RGB values for all yellow stripe pixels in frog's
vent


    r=feval(eqR,frf); g=feval(eqG,fgf); b=feval(eqB,fbf);
    rgb=[r g b]; tot=sum(rgb,2); rgb=rgb./tot; %toral relative
intensities for RGB
    mnsf=mean(rgb); sdsf=std(rgb); mlumf=mean(tot); slumf=std(tot);  %all
frog

    r=feval(eqR,frh); g=feval(eqG,fgh); b=feval(eqB,fbh);
    rgb=[r g b]; tot=sum(rgb,2); rgb=rgb./tot; %toral relative
intensities for RGB
    mnsh=mean(rgb); sdsh=std(rgb); mlumh=mean(tot); slumh=std(tot);
%head

    r=feval(eqR,frt); g=feval(eqG,fgt); b=feval(eqB,fbt);
    rgb=[r g b]; tot=sum(rgb,2); rgb=rgb./tot; %toral relative
intensities for RGB
    mnst=mean(rgb); sdst=std(rgb); mlumt=mean(tot); slumt=std(tot);
%torso

    r=feval(eqR,frv); g=feval(eqG,fgv); b=feval(eqB,fbv);
    rgb=[r g b]; tot=sum(rgb,2); rgb=rgb./tot; %toral relative
intensities for RGB
    mnsv=mean(rgb); sdsv=std(rgb); mlumv=mean(tot); slumv=std(tot);
%vent


    fprintf(fid,'%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f',...
        mnsf(1),mnsf(2),mnsf(3),mlumf,sdsf(1),sdsf(2),sdsf(3),slumf);
    fprintf(fid,',%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f',...
        mnsh(1),mnsh(2),mnsh(3),mlumh,sdsh(1),sdsh(2),sdsh(3),slumh);
    fprintf(fid,',%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f',...
        mnst(1),mnst(2),mnst(3),mlumt,sdst(1),sdst(2),sdst(3),slumt);
    fprintf(fid,',%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f,%6.4f',...
        mnsv(1),mnsv(2),mnsv(3),mlumv,sdsv(1),sdsv(2),sdsv(3),slumv);
     fprintf(fid,',%s\n',fname);
     fprintf(1,'Photo %s finished\n',fname);
  end; %if v>0
 end; %file selected or v=1 while loop
 fclose(fid);
 fprintf(1,'Finished, results in %s\n',oname);
end %function



function [dirs,nds]=GetDIRList(dirname);
% [dirs,nds]=GetDIRList(dirname);
% INPUT (string in single-quotes)
%   dirname is a directory name, such as 'C:\Active\Heinsohn\'  % Must
end '\'
% OUTPUT
%   dirs is a list of the directories within dirn (without dirn in front)
%   nds    is the number of directories (can be 0)
drl=dir(dirname); [sz,d]=size(drl);
dirs=[]; nds=0;
for d=1:sz
   temp=drl(d).name;
   typ =drl(d).isdir;
   k1=findstr('.',temp); if isempty(k1) k1=0; end;
   k2=findstr('..',temp); if isempty(k2) k2=0; end;
   if ((typ==1) & (k1==0) & (k2==0))
       dirs=[dirs; cellstr(temp)];
       nds=nds+1;
   end;
end;
end %function

function [files,nf]=GetFileList(dirn,stype)
% [files,nf]=GetFileList(dirn,stype);
% INPUT (strings in single-quotes)
%   dirn is a directory name, such as 'C:\Active\Heinsohn\'  % Must end
'\'
%    to read in the current directory, use '' (empty)
%   stype is the kind of file, such as '.ttt' or '.Master.transmission'
% OUTPUT
%   files is a list of the files (without the directory in front)
%   nf    is the number of files (can be 0)
% Note: will recognize .Master.tra with stype='.tra'
drp=[dirn '*' stype];
drl=dir(drp); [nf,f]=size(drl);
files=[];
for f=1:nf
  temp=drl(f).name;
  files=[files; cellstr(temp)];
end;
end %function

function [pcx,pcy]=polycenter(px,py);
% [pcx,pcy]=polycenter(px,py); finds the geometric center (pcx,pcy)
%   of a polygon whose points are in (px,py) column vectors
[np,i]=size(px);
for j=1:np
```

```
    i=j+1;  if j==np i=1; end;
    a(j)=((py(i)+py(j))*(px(j)-px(i)))/2;
    my(j)=(((py(i)^2)+py(i)*py(j)+(py(j)^2))*(px(j)-px(i)))/6;
    mx(j)=(((px(i)^2)+px(i)*px(j)+(px(j)^2))*(py(j)-py(i)))/6;
end;
sx=0; sy=0; sa=0;
for j=1:np
    sa=sa+a(j);
    sx=sx+mx(j);
    sy=sy+my(j);
end;
pcx=-sx/sa;
pcy=sy/sa;
end %function


function GetHueChrLum;
%calculate Hue, Chroma and Luminance from calibrated RGB from photos
clear;
[fname,path,f]=uigetfile('.csv','Select a CSV RGB file');

oname=fname; k=strfind(oname,'.'); if isempty(k) k=0; end;
oname=[oname(1:(k-1)) 'WithHueChrLum' oname(k:end)];
ofid=fopen(oname,'wt');


fid=fopen(fname);
rd=textscan(fid,'%s',1,'whitespace','\n'); head1=char(rd{1});
rd=textscan(fid,'%s %s %s %s %s %s %s %s %s',1,'delimiter',',');
heads=[rd{1:9}];
rd=textscan(fid,'%f %f %f %f %f %f %f %f %s','delimiter',',');
fclose(fid);
head1=regexprep(head1,'"','');
head1=regexprep(head1,',','');
R=rd{1}; G=rd{2}; B=rd{3}; Lum=rd{4}; sdR=rd{5}; sdG=rd{6}; sdB=rd{7};
sdTot=rd{8};
photo=rd{9}; n=length(R); clear fid rd;
Hue=zeros(n,1); Chr=Hue;
for k=1:n
  r=R(k); g=G(k); b=B(k); [cx,cy]=PXYZtoTRI(1/3,1/3,1/3);
  [tx,ty]=PXYZtoTRI(r,g,b); tx=tx-cx; ty=ty-cy; %set gray to (0,0)
  [hue,chr]=cart2pol(ty,tx); Hue(k)=rad2deg(hue); %red -120, green 0,
blue +120
  Chr(k)=chr/0.5774;  %maximum chroma set to 1
end;

fprintf(ofid,'%s\n',head1);
fprintf(ofid,'Hue,Chroma,Luminance');
for k=1:9 fprintf(ofid,',%s',heads{k}); end;
fprintf(ofid,'\n');
for k=1:n
   fprintf(ofid,'%8.4f,%6.4f,%6.4f',Hue(k),Chr(k),Lum(k));

fprintf(ofid,',%8.4f,%8.4f,%8.4f,%8.4f,%8.4f,%8.4f,%8.4f,%8.4f,%s\n',...
        R(k),G(k),B(k),Lum(k),sdR(k),sdG(k),sdB(k),sdTot(k),photo{k});
end;
fclose(ofid);
fprintf(1,'Data read from %s\n',fname);
fprintf(1,'Results in     %s\n',oname);

end %main function

function [tx,ty]=PXYZtoTRI(p1,p2,p3);
%[tx,ty]=PXYZtoTRI(p1,p2,p3);  converts proportions p1,p2,p3
%   to triangular coordinates tx,ty        (sets p1+p2+p3=1)
%   Triangle edge lengths 1.0   [height sqrt(3)/2]
%   Triangle vertices: (0, 0), (1/2, sqrt3/2), (1, 0)

% first ensure p2+p3+p1 = 1
[n,i]=size(p2);
for i=1:n
    tot=p1(i)+p2(i)+p3(i);
    if tot>0  p1(i)=p1(i)/tot; p2(i)=p2(i)/tot; p3(i)=p3(i)/tot; end;
end;
% convert to triangle
s3=sqrt(3);
for i=1:n
    tx(i)=(p2(i)+2*p3(i))/s3; ty(i)=p2(i);
end;
%rescale so that triangle edges have length=1
for i=1:n
    tx(i)=tx(i)*s3/2; ty(i)=ty(i)*s3/2;
end;
tx=tx'; ty=ty'; %convert to column vectors;
end %triangle
```