Supplementary information

Strategy for optimizing experimental settings for studying low atomic number colloidal assemblies using liquid phase scanning transmission electron microscopy

Peter Kunnas, Mohammad-Amin Moradi, Nico Sommerdijk, Niels de Jonge*

Contents

SI 1.1 Buffer solutions

Solution	NaH_2PO_4 (μl)	NaOH (μl)	HCl (μl)	NaCl (μl)
aline	-	-	-	440
PBS pH=2	100	-	120	470
PBS pH=7	100	50	-	440
PBS pH=12	100	320	-	-

Table S1. Amounts of 1M stock solutions mixed to prepare 0.01 M phosphate-buffered saline (PBS) with a total volume of 10 ml.

SI 2.1 MATLAB script for calculating the attainable resolution

To calculate the attainable resolution (d) for bright-field (BF) and dark-field (DF) STEM, this MATLAB script uses the theory from:

N. de Jonge, Theory of the spatial resolution of (scanning) transmission electron microscopy in liquid water or ice layers, Ultramicroscopy 187 (2018) 113.

Equation numbers refer to the same publication unless corrections have been given in the Corrigendum:

de Jonge N. Corrigendum to: "Theory of the spatial resolution of (scanning) transmission electron microscopy in liquid water or ice layers" [Ultramicroscopy 187 (2018) 113-125]. Ultramicroscopy. 2019 Jan;196:129-130. doi: 10.1016/j.ultramic.2018.10.006. Epub 2018 Oct 18. Erratum for: Ultramicroscopy. 2018 Apr;187:113-125. PMID: 30366317.

Simulation settings

```
clear

%mode="BF_STEM";
mode="DF_STEM";
%inelastic="off"
inelastic="on"
%object_material="Au"
%object_material="C"
%object_material="SiO2"
object_material="C8H8"
t=0.3E-6 %Liquid thickness
D=10E20 %electron dose e- per m^2
d=30E-9 %Object diameter
z=0 %Object distance from the top window (STEM) or bottom window(TEM)
t_SiN=50E-9
```

Sample parameters

Parameter	symbol	Value	Unit	Comment
Atomic number	Z			
Atomic weight	W		$kg\ mol^{-1}$	
Density	ρ		$kg\ m^{-3}$	
Total sample thickness	x		m	
Window thickness	t_{SiN}		m	
Liquid thickness	t		m	
Window thickness	t_{SiN}		m	
Object thickess	d		m	
Object distance from the window	z		m	Top window

```
Z_H=1;Z_C=6;Z_N=7;Z_O=8;Z_Si=14;Z_Au=79;
Z_H2O=sqrt(0.67*Z_H^2+0.33*Z_O^2)
Z_C8H8=sqrt(0.5*Z_C^2+0.5*Z_H^2)
Z_SiO2=sqrt(0.33*Z_Si^2+0.67*Z_O^2)
Z_SiN=sqrt(3/7*Z_Si^2+4/7*Z_N^2)
W_H=1.008E-3;W_C=12.01E-3;W_N=14E-3;W_O=16E-3;W_Si=28.08E-
3;W_Au=196.97E-3;
W_H2O=2/3*W_H+1/3*W_O
W_SiO2=1/3*W_Si+2/3*W_O
W_SiN=3/7*W_Si+4/7*W_N
W_C8H8=0.5*W_C+0.5*W_H
rho_H2O=1000;
rho_SiO2=1630;rho_C=2250;rho_SiN=3170;rho_Au=19300;rho_C8H8=1050;
Z=1;theta_0=1;
```

Constants and Variables defined

Constant	symbol	Value	Unit	Comment
Planck's constant	h	$6.62607004 \times 10^{-34}$	m^2kg/s	
Speed of light	c	2.99792458×10^{8}	m/s	
Electron rest mass	m_0	$9.10938370 \times 10^{-31}$	kg	
Bohr radius	a_h	5.29×10^{-11}	m	
Permittivity of the space	ϵ_0	8.85×10^{-12}	F/m	
Avogadro's number	N_a	6.02214×10^{23}	mol^{-1}	
Elementary charge	e	1.602×10^{-19}	C	$J\ eV^{-1}$

```
h=6.62607004E-34
c=2.99792458E8
m_0=9.10938370E-31
a_h=5.29E-11
epsilon_0=8.85E-12
N_a=6.02214E23
e=1.6021766E-19
```

Variable	symbol	Value	Unit	Comment
Partial el. scattering cross-section	$\sigma_{el}(\theta)$		m^2	Eq.1
Scattering angle	θ		rad	
Atomic number	Z			
Relativistic wave length	λ		m	Eq.2
Electon energy	E		eV	
Electon rest energy	E_0		eV	Eq.3
Characteristic scattering angle	θ_0		rad	Eq.4
Partial Inel. scattering cross-section	$\sigma_{inel}(\theta)$		nm^2	Eq.5
Total cross-section	$\sigma(\theta)$		nm^2	Eq.6
DF scattering contrast	N/N_0			Eq.7
Specimen thickness	t		m	
Mean-free-path length	$l(\theta)$		m	Eq.8
Partial el. scattering cross-section of water			nm^2	Eq.9
BF scattering contrast	M/M_0			Eq.10
Broadened probe diameter	d_{blur}		m	Eq.11 and 12
Diffraction limited probe diameter	d_{diff}		m	Eq.13
Spher.Aberr. limited probe diameter	d_{CS}		m	Eq.14
Probe diameter containing 50% of current	d_{50}		m	Eq.15
Electrons scattered on DF detector by the sample	N_{signal}			Eq.16
Electrons scattered on DF detector by the background	N_{signal}			Eq.17
Signal-to-noise ratio	SNR			Eq.18, Rose criterion =3
Electron dose	D		$e^{-}m^{-2}$	Eq.21
Noise-limited resolution	d_{SNR}			Eq.24
Electrons scattered on BF detector by the sample	M_{signal}			Eq.25
Electrons scattered on BF detector by the background	M_{signal}			Eq.26
Beam blurring dialmeter in BF-STEM	d_{blur_BF}			Eq.28
Spatial resolution for STEM	d_{STEM}			Eq.29
Scherzer resolution	d_{Sch}			Eq.30
Electron scattering factor	$f(\theta)$			Eq.30
Differential scattering cross-ection	$d\sigma/d\Omega$			Eq.30
El. Differential scattering cross-ection	$\sigma_{el}/d\Omega$			Eq.32
Elastic zero-angle scattering sattering factor	$f_{el}(0)$			Eq.33
Contrast	C			Eq.34
SNR for TEM phase contrast	SNR_{phase}			Eq.35
Dose-limited resolution for TEM phase contrast	d_{SNR_phase}		m	Eq.36
Broadening of the energy spread	$\Delta E(x)$			Eq.37
Total sample thickness	x		m	
Relativistic velocity	v		m/s	Eq.38
Chrom. aber.contriburion on resolution	d_{cc}		m	Eq.39
Chrom. aber. limited resolution for TEM	d_{TEM_cc}		m	Eq.41
SNR-limited resolution for TEM	d_{SNR}		m	Eq.41
Spatial resolution for TEM	d_{TEM}			Eq.42
Mean-free-path length in water	l_w		m	
Mean-free-path length in object	l_o		m	
Mean-free-path length in window	l_{SiN}		m	

Microscope parameters

Parameter	symbol	Value	Unit	Comment
Electron dose	D		$e^{-}\ m^{-2}$	Eq.21
pixel size	s		m	
Acceleration voltage	U		eV	
Electon energy	E		J	
STEM probe convergence semi-angle	α_p		rad	
TEM objective lens aperture semi-angle	α		rad	
STEM BF outer collection-semi-angle	β_{BF_STEM}		rad	
STEM DF inner collection-semi-angle	β_{DF_STEM}		rad	
Spherical aberration for STEM	$c_{s(STEM)}$		m	
Spherical aberration for TEM	$c_{s(TEM)}$		m	
Chromatic aberration for TEM	c_{cc}		m	
Chromatic aberration for TEM	c_{cc}		m	
Energy spread for the electron source	ΔE_{source}		eV	

```
s=0.48E-9
U=2E5
E=U*e
```

```
cs_STEM=0.5E-3
```

Adjustment of the probe semi-angle (α)

BF

```
if mode=="BF_STEM"
    beta_BF_STEM=[1E-3:1E-3:0.1];
    alpha_p_BF_STEM=zeros(1,100);

    for ii = 1:length(beta_BF_STEM)
        if beta_BF_STEM(ii)>8E-3
            alpha_p_BF_STEM(ii) = 8E-3;
        else
            alpha_p_BF_STEM(ii) =beta_BF_STEM(ii);
        end
    end
```

```
    alpha_p=alpha_p_BF_STEM
    theta=beta_BF_STEM
    beta=beta_BF_STEM
    c_s=cs_STEM
    alpha_TEM=theta %for testing purposes
end
```

DF

```
if mode=="DF_STEM"
    beta_DF_STEM=[1E-3:1E-3:0.1];
    alpha_p_DF_STEM = zeros(1,100);
    for ii = 1:length(beta_DF_STEM)
        if beta_DF_STEM(ii)>16E-3
            alpha_p_DF_STEM(ii) = 8E-3;
        else
            alpha_p_DF_STEM(ii) =beta_DF_STEM(ii)/2;
        end
    end
    alpha_p=alpha_p_DF_STEM
    theta=beta_DF_STEM
    beta=beta_DF_STEM
    c_s=cs_STEM
    alpha_TEM=theta %for testing purposes
end
```

Electron rest energy (Eq. 3)

```
symstr="E_0=m_0*c^2";
displayFormula symstr;
E_0=m_0*c^2
```

Relativistic wavelength (Eq. 2)

```
symstr="lambda=h*c/(sqrt(2*E*E_0+E^2))";
```

```
displayFormula symstr
lambda=h*c/sqrt((2*E*E_0+(E)^2))
```

Characteristic angle

```
symstr="theta_0=(lambda*Z^(1/3))/(2*pi*a_h)"
displayFormula symstr

theta_0_H=(lambda*Z_H^(1/3))/(2*pi*a_h)
theta_0_C=(lambda*Z_C^(1/3))/(2*pi*a_h)
theta_0_N=(lambda*Z_N^(1/3))/(2*pi*a_h)
theta_0_O=(lambda*Z_O^(1/3))/(2*pi*a_h)
theta_0_Si=(lambda*Z_Si^(1/3))/(2*pi*a_h)
theta_0_Au=(lambda*Z_Au^(1/3))/(2*pi*a_h)
```

Elastic scattering (Eq. 1)

"The absolute value of σel or its reciprocal xel = A/(NAσel) does not agree well with experiments (Fig. 6.3) because the Wentzel screening model (5.22) is too simple and the Born approximation fails for high Z" Reimer pp.153

```
symstr="sigma_el=(1/pi)*Z^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0)^2)";
displayFormula symstr

sigma_el_H=(1/pi)*Z_H^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_H).^2)
sigma_el_C=(1/pi)*Z_C^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_C).^2)
sigma_el_N=(1/pi)*Z_N^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_N).^2)
sigma_el_O=(1/pi)*Z_O^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_O).^2)
sigma_el_Si=(1/pi)*Z_Si^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_Si).^2)
sigma_el_Au=(1/pi)*Z_Au^(4/3)*lambda^2*(1+E/E_0)^2./(1+(theta./theta_0_Au).^2)
```

Comparison from Nist database: For 200kV

Au: 1.439E-21 m^2

C: 5.004 E-23

O: 5.68252E-23

H: 2.260864 E-24

```
clf
legend
hold on
plot(beta,sigma_el_H,'--r','DisplayName','\sigma_{el\_H}');
plot(beta,sigma_el_C,'--g','DisplayName','\sigma_{el\_C}');
plot(beta,sigma_el_O,'--b','DisplayName','\sigma_{el\_O}');
plot(beta,sigma_el_Si,'--c','DisplayName','\sigma_{el\_Si}');
plot(beta,sigma_el_Au,'--m','DisplayName','\sigma_{el\_Au}');
hold off
```

Inelastic scattering (Eq.5)

For theta >0 rad

```
symstr="sigma_inel=(4/pi)*Z^(1/3)*lambda^2*(1+E/E_0)^2*(-
1/(4*(1+(theta/theta_0)^2))+log(sqrt(1+(theta_0/theta)^2)))";
displayFormula symstr
```

For theta = 0 rad (Total inelastic scattering crossection) (reimer pp.163)

```
symstr="sigma_inel=26/Z*sigma_el";
displayFormula symstr
```

```
sigma_inel_H=(4/pi)*Z_H^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_H).^2))+log(sqrt(1+(theta_0_H./theta).^2)));
%sigma_inel_H(1)=26/Z_H*sigma_el_H(1)

sigma_inel_C=(4/pi)*Z_C^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_C).^2))+log(sqrt(1+(theta_0_C./theta).^2)));
%sigma_inel_C(1)=26/Z_C*sigma_el_C(1)
sigma_inel_N=(4/pi)*Z_N^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_N).^2))+log(sqrt(1+(theta_0_N./theta).^2)));
%sigma_inel_N(1)=26/Z_N*sigma_el_N(1)
```

```
 sigma_inel_O=(4/pi)*Z_C^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_C).^2))+log(sqrt(1+(theta_0_C./theta).^2)));
 %sigma_inel_O(1)=26/Z_O*sigma_el_O(1)
 sigma_inel_Si=(4/pi)*Z_Si^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_Si).^2))+log(sqrt(1+(theta_0_Si./theta).^2)));
 %sigma_inel_Si(1)=26/Z_Si*sigma_el_Si(1)
 sigma_inel_Au=(4/pi)*Z_H^(1/3)*lambda^2*(1+E/E_0)^2*(-
1./(4*(1+(theta./theta_0_Au).^2))+log(sqrt(1+(theta_0_Au./theta).^2)));
 %sigma_inel_Au(1)=26/Z_Au*sigma_el_Au(1)
```

Fraction (f) of R of elastic and inelastic scattering

```
 v_H=sigma_inel_H./(sigma_el_H);
 v_C=sigma_inel_C./(sigma_el_C);
 v_O=sigma_inel_O./(sigma_el_O);
 v_Si=sigma_inel_Si./(sigma_el_Si);
 v_Au=sigma_inel_Au./(sigma_el_Au);

 clf
 legend
 hold on
 plot (beta,v_H,'-r','DisplayName','v_H');
 plot (beta,v_C,'-g','DisplayName','v_C');
 plot (beta,v_O,'-b','DisplayName','v_O');
 plot (beta,v_Si,'-c','DisplayName','v_{Si}');
 plot (beta,v_Au,'-m','DisplayName','v_{Au}');
 hold off
```

Comment: Reimer gives a value for v=~26/Z to 20/Z at theta=0 (Total inelastic cross-section)

Total scattering cross-section:

```
 if inelastic=="off"
 symstr="sigma(theta)=sigma_el(theta)";
 displayFormula symstr
 sigma_H=sigma_el_H;
 sigma_C=sigma_el_C;
 sigma_N=sigma_el_N;
```

```
sigma_O=sigma_el_O;
sigma_Si=sigma_el_Si;
sigma_Au=sigma_el_Au

else
symstr="sigma(theta)=sigma_el(theta)+sigma_inel(theta)";
displayFormula symstr
sigma_H=sigma_inel_H+sigma_el_H;
sigma_C=sigma_inel_C+sigma_el_C;
sigma_N=sigma_inel_N+sigma_el_N;
sigma_O=sigma_inel_O+sigma_el_O;
sigma_Si=sigma_inel_Si+sigma_el_Si;
sigma_Au=sigma_inel_Au+sigma_el_Au
end



clf
legend
hold on
plot(beta,sigma_el_H,'--r','DisplayName','\sigma_{el\_H}');
plot(beta,sigma_el_C,'--g','DisplayName','\sigma_{el\_C}');
plot(beta,sigma_el_O,'--b','DisplayName','\sigma_{el\_O}');
plot(beta,sigma_el_Si,'--c','DisplayName','\sigma_{el\_Si}');
plot(beta,sigma_el_Au,'--m','DisplayName','\sigma_{el\_Au}');

%plot (beta,sigma_H,'-r','DisplayName','\sigma_H');
plot (beta,sigma_C,'-g','DisplayName','\sigma_C');
plot (beta,sigma_O,'-b','DisplayName','\sigma_O');
plot (beta,sigma_Si,'-c','DisplayName','\sigma_{Si}');
plot (beta,sigma_Au,'-m','DisplayName','\sigma_{Au}');

hold off
```

The effect of inelastic scattering is small beyond few mrad

Mean free path length

Combined cross-section for water

```
symstr="sigma_H2O=0.67*sigma_H+0.33+sigma_O";
displayFormula symstr
sigma_H2O=0.67*sigma_H+0.33*sigma_O;
```

Combined cross-section for Silica

```
symstr="sigma_SiO2=0.33*sigma_Si+0.67*sigma_O";
displayFormula symstr
sigma_SiO2=0.33*sigma_Si+0.67*sigma_O;
```

Combined cross section for SiN

```
symstr="sigma_SiN=0.75*sigma_Si+1.3*sigma_N";
displayFormula symstr
sigma_SiN=0.75*sigma_Si+1.3*sigma_N;
```

Combined cross section for polystyrene

```
symstr="sigma_C8H8=0.5*sigma_C+0.5*sigma_H";
displayFormula symstr
sigma_C8H8=0.5*sigma_C+0.5*sigma_H;

symstr="l=W/(sigma_el*rho*N_a)";
displayFormula symstr

l_H2O=W_H2O./(sigma_H2O.*rho_H2O*N_a);
l_C8H8=W_C8H8./(sigma_C8H8.*rho_C8H8*N_a);
l_SiO2=W_SiO2./(sigma_SiO2.*rho_SiO2*N_a);
l_SiN=W_SiN./(sigma_SiN.*rho_SiN*N_a);
l_C=W_C./(sigma_C.*rho_C*N_a);
l_Au=W_Au./(sigma_Au.*rho_Au*N_a);




clf
```

```
legend
hold on
%plot(beta,l_el_H2O,'.b','DisplayName','l_{el\_H2O}');
%plot(beta,l_inel_H2O,'--b','DisplayName','l_{inel\_H2O}');
%plot(beta,l_H2O,'-b','DisplayName','l_{H2O}');
clf
hold on
plot(beta,l_H2O,'-b','DisplayName','l_{H2O}');
plot(beta,l_SiO2,'-m','DisplayName','l_{SiO2}');
plot(beta,l_SiN,'-g','DisplayName','l_{SiN}');
plot(beta,l_C,'-k','DisplayName','l_C');
plot(beta,l_Au,'-','color',[0.9290 0.6940 0.1250]','DisplayName','l_{Au}');
hold off
```

Contrast DF

```
symstr="N/N_0=1-exp(1)^(l/t)";
displayFormula symstr
symstr="N/N_0=N_0(1-exp(1)^(l/t))";
displayFormula symstr
%C=1-exp(1).^(-t./l)
%plot (C)
hold on
```

Contrast BF

```
symstr="C=N/N_0=exp(1)^(-t/l)";
displayFormula symstr
%C=exp(1).^(-t./l)
%plot (C)
%hold off
```

Beam broadening (Eq. 11)

```
symstr="d_blur=1.3*(lambda^2/(2*pi*a_h))*z^1.5*sqrt((N_a*rho)/(3*pi*W))*Z(1+E/E_0)";
displayFormula symstr
d_blur_H2O=blur(a_h, lambda, N_a, E_0, E, W_H2O, Z_H2O, rho_H2O, z); %blurring
from the liquid
```

```
 d_blur_SiN=blur(a_h, lambda, N_a, E_0, E, W_SiN, Z_SiN, rho_SiN, t_SiN); %blurring from the window
 symstr="d_blur=d_blur_SiN(t_SiN)+d_blur_H2O(z)";
 displayFormula symstr
 if mode=="DF_STEM"
    d_blur=double.empty(); %array for output
    for col=1:size(beta,2) %beta variation
       d_blur=cat(2,d_blur,(d_blur_SiN+d_blur_H2O)); %output an array of constant value for each beta in DF_STEM;
    end
 d_blur
 end

 if mode=="BF_STEM"
    d_blur=double.empty() %array for output
    mode

    if z==0
       z
       for col=1:size(beta,2) %beta variation
          d_blur=cat(2,d_blur,(d_blur_SiN)); %output an array of constant value for each beta in DF_STEM. Contribution coming from the window only when z=0;
       end
       d_blur
    end


    if z>0
       z
       theta_50= atan((d_blur_SiN+d_blur_H2O)/(z)) %Calculate the semi-angle of the cone when it exits the sample
       for col=1:size(beta,2) %beta variation
          if theta_50<beta(col)
             d_blur_BF=tan(theta_50)*z;
             d_blur=cat(2,d_blur, d_blur_BF+d_blur_SiN);
          end
```

```
            if theta_50>=beta(col);
                d_blur_BF=tan(beta(col))*z;
                d_blur=cat(2,d_blur, d_blur_BF+d_blur_SiN);
            end
        end
        d_blur
    end
end

if mode=="BF_TEM" %TEM corresponds to BF-STEM with point detector
    d_blur=double.empty() %array for output
    mode

    if z==0

        for col=1:size(beta,2) %beta variation
            d_blur=cat(2,d_blur,(d_blur_SiN)); %output an array of constant value for each
beta in DF_STEM. Contribution coming from the window only when z=0;
        end
    end



    if z>0

        theta_50= atan((d_blur_SiN+d_blur_H2O)/(z)) %Calculate the semi-angle of the
cone when it exits the sample
        for col=1:size(beta,2) %beta variation
            d_blur_BF=tan(theta_50)*z;; %TEM corresponds to BF-STEM with point
detector and hence theta_50>=beta(col);
            d_blur=cat(2,d_blur, d_blur_BF+d_blur_SiN);
        end

    end

end
```

```
symstr="d_diff=0.54*(lambda/alpha_p)";
displayFormula symstr
d_diff=0.54*(lambda./alpha_p);
clf
hold on
plot(alpha_p,d_diff,'-k','DisplayName','d_diff');
hold off

symstr="d_cs=2^(-5/2)*c_s*(alpha_p)^3";
displayFormula symstr
d_cs=2^(-5/2)*c_s*(alpha_p).^3
clf
hold on
plot(alpha_p,d_cs,'k','DisplayName','d_cs');
hold off
symstr="d_50=sqrt(d_diff^2+d_cs^2)";
displayFormula symstr
d_50=sqrt(d_diff.^2+d_cs.^2)
clf
hold on
plot(alpha_p,d_50,'-k','DisplayName','d_50');
hold off
```

SNR-limited resolution for DF-STEM

Solving Eq. 16 and 17 numerically

```
%Solving equations 16 and 17 numerically

symstr="N_signal=N_0*(1-exp(1)^(-(d_min/l_o+(t-d_min)/l_H2O+2*t_SiN/l_SiN)))";

displayFormula symstr
symstr="N_bkg=N_0*(1-exp(1)^(-(t/l_H2O+2*t_SiN/l_SiN)));";

displayFormula symstr

if object_material=="Au"
    l_o=l_Au
```

```
end

if object_material=="C"
    I_o=I_C
end

if object_material=="SiO2"
    I_o=I_SiO2
end

if object_material=="C8H8"
    I_o=I_C8H8
end
[d_SNR, N_o, N_bg]=d_snr_num(mode,beta, D, I_o, I_H2O, I_SiN, t, t_SiN) %This calls for the function that calculated the d_SNR. In addition, it returns the number of electrons hitting the detetor when the beam is on the location of a nano-object(N_o) or the liquid background (N_bg) which were used to show the contrast inversion for PMS.
N=N_o./N_bg %This reports the sign of contrast when N passes 1
clf
hold on
%set(gca, 'YScale', 'log')
plot(beta,N,'-k','DisplayName','N');
%plot(beta,N_o,'-k','DisplayName','N');
%plot(beta,N_bg,'-k','DisplayName','N');

hold off
```

```
symstr="d_STEM=sqrt((2*d_50).^2+d_SNR.^2+d_blur.^2)";
displayFormula symstr
d_STEM=sqrt((2*d_50).^2+d_SNR.^2+d_blur.^2)

%txt=['Resolution']
%text(4,0.5,txt)
```

```
 export_array_d=double.empty(); %array for exporting the data
 export_array_d=cat(1,export_array_d,beta*1E3); %export values in mrad

 clf
 legend
 hold on


 if mode=="BF_STEM"
    title(['Resolution BF-STEM ', object_material])
    plot(beta*1E3, d_50*1E9,'--r','DisplayName','d_{50}');
    plot(beta*1E3, d_SNR*1E9,'--m','DisplayName','d_{SNR}');
    plot(beta*1E3, d_blur*1E9,'--b','DisplayName','d_{blur}');
    plot(beta*1E3, d_STEM*1E9,'.k','DisplayName','d_{STEM}');

    export_array_d=cat(1,export_array_d,d_STEM*1E9) % Prepares data for export, units
changed to mrads and nanometers
    export_array_d=cat(1,export_array_d,d_SNR*1E9)
    export_array_d=cat(1,export_array_d,d_blur*1E9)
    export_array_d=cat(1,export_array_d,d_50*1E9)



    format_names= ["%12s", "%12s", "%12s", "%12s", "%12s\n"]
    column_names = ["Beta", "d_STEM_BF", "d_SNR", "d_blur", "d_50"]
    format_values =["%12.12f","%12.12f","%12.12f","%12.12f", "%12.12f\n"]
    format_units=["%12s", "%12s", "%12s", "%12s", "%12s\n"]
    column_units=["mrad", "nm", "nm", "nm", "nm"]
    [M,I]=min(d_STEM) %minimum resolution found M at the indice I

 end

 if mode=="DF_STEM"
    title(['Resolution DF-STEM ', object_material])
    plot(beta*1E3, d_50*1E9,'--r','DisplayName','d_{50}');
    plot(beta*1E3, d_SNR*1E9,'--m','DisplayName','d_{SNR}');
    plot(beta*1E3, d_blur*1E9,'--b','DisplayName','d_{blur}');
    plot(beta*1E3, d_STEM*1E9,'.b','DisplayName','d_{STEM}');
```

```
    export_array_d=cat(1,export_array_d,d_STEM*1E9) % Prepares data for export, units
changed to mrads and nanometers
    export_array_d=cat(1,export_array_d,d_SNR*1E9)
    export_array_d=cat(1,export_array_d,d_blur*1E9)
    export_array_d=cat(1,export_array_d,d_50*1E9)



    format_names= ["%12s", "%12s", "%12s", "%12s", "%12s\n"]
    column_names = ["Beta", "d_STEM_DF", "d_SNR", "d_blur", "d_50"]
    format_values =["%12.12f","%12.12f","%12.12f","%12.12f", "%12.12f\n"]
    format_units=["%12s", "%12s", "%12s", "%12s", "%12s\n"]
    column_units=["mrad", "nm", "nm", "nm", "nm"]

    [M,I]=min(d_STEM) %minimum resolution found M at the indice I
 end

 parameters=["Mode=",mode, "Object material=",object_material, "t=",num2str(t*1E9),
"t_SiN=", t_SiN*1E9, "De=",num2str(D*1E-20), "z=", num2str(z*1E9), "Inelastic=",
inelastic, "Optimized_d", M*1E9,"at_beta", beta(I)*1E3]
 format_parameters=["%12s","%12s\n","%12s","%12s\n","%12s","%12f\n","%12s","%12f\
n",
"%12s","%12f\n","%12s","%12f\n","%12s","%12s\n","%12s","%12f\n","%12s","%12f\n"]

 filename=[mode,"_",object_material,"_t=",num2str(t*1E9),
"_t_SiN=",num2str(t_SiN*1E9) ", _De=",num2str(D*1E-20), "_z=", num2str(z*1E9),
"_inelastic=", inelastic]
 filename_data=join([filename, ".txt"],"")
 fileID = fopen(filename_data,'w+');
 fprintf(fileID,join(format_parameters), parameters);
 fprintf(fileID,join(format_names), column_names);
 fprintf(fileID,join(format_units), column_units);
 fprintf(fileID,join(format_values),export_array_d);
 fclose(fileID)

 xlabel('beta (mrad)')
 ylabel('d(nm)')
```

```
 xlim([0 100])
 ylim([0 200])


 dim = [.2 .5 .3 .3];
 str = ['t=',num2str(t*1E9), ' nm ' ,' t_{SiN=}', num2str(t_SiN*1E9), ' nm', ' D_e=',
num2str(D*1E-20), ' e^-Å^{-2}', ' z=', num2str(z*1E9), ' nm'];
 annotation('textbox',dim,'String',str,'FitBoxToText','on');

 text(beta(l)*1E3,M*0.8E9,['\uparrow d=', num2str(M*1E9), ' nm at \beta ='
num2str(beta(l)*1E3), 'mrad'])
 saveas(gcf,join(filename,""),'png')
```

```
 function d_blur=blur(a_h, lambda, N_a, E_0, E, W, Z, rho, z)
 d_blur=1.5/2*(lambda^2/(2*pi*a_h))*(z)^1.5*sqrt((N_a*rho)/(3*pi*W))*Z*(1+E/E_0);
 %corrigendum de Jonge
 end


 function sigma_el=sigma(a_h, theta, lambda, E, theta_0, E_0)

 end

 function sigma_el(a_h, theta, lambda, E, Z, theta_0, E_0)
 sigma_el=Z.^2*(a_h*Z.^(-
1/3)).^2*lambda.^2*(1+E/E_0).^2./pi*a_h.^2./(1+(theta./theta_0).^2)
 end


 %d_snr_num solves the snr-limited resolution numerically
 function [d_snr_num, N_o, N_bg]=d_snr_num(mode, beta, D, l_o, l_H2O, l_SiN, t, t_SiN
)

 syms d_min %variable to be solved numerically
 output_array_d_min=double.empty(); %array for output
 output_array_N_o=double.empty();
```

```
 output_array_N_bg=double.empty();

 if mode=="DF_STEM" %When darkfield is selected

 N_0=D*d_min^2/4;

 for col=1:size(beta,2) %beta variation
    N_signal=N_0*(1-exp(1)^(-(d_min/l_o(col)+(t-
d_min)/l_H2O(col)+2*t_SiN/l_SiN(col))));
    N_bkg=N_0*(1-exp(1)^(-(t/l_H2O(col)+2*t_SiN/l_SiN(col))));
    SNR=abs(N_signal-N_bkg)/sqrt(N_bkg)==3;% Absolute value of the signal taken as
the contrast can be negative or positive
    num_sol = vpasolve(SNR, d_min); %numerical solution by using VPASOLVE;

    if isempty(num_sol)==1
       output_array_d_min=cat(2,output_array_d_min,NaN);
    end

    if isempty(num_sol)==0
       output_array_d_min=cat(2,output_array_d_min,double(num_sol));
    end
    % The six following lines use the numerically solved d_min to calculate the number of
electrons that hit the detector when the probe is located on an object  (N_o and N_bk,
respectively.)
    % This was done to double check that the negative values of d_SNR_min  obtained
from the numerical solution are indeed due to the contrast inversion.
    num_sol=abs(num_sol); % Six following lines use the numerically solved d_min to
calculate the number of electrons that hit the detector when the probe is located on an
object  (N_o and N_bk, respectively.)
    N_0=D*num_sol^2/4; %This was done to double check that the negative values of
d_SNR_min  obtained from the numerical solution are in deed due to the contrast
inversion.
    N_o=N_0*(1-exp(1)^(-(num_sol/l_o(col)+(t-
num_sol)/l_H2O(col)+2*t_SiN/l_SiN(col))));
    output_array_N_o=cat(2,output_array_N_o,double(N_o));
    N_bg=N_0*(1-exp(1)^(-(t/l_H2O(col)+2*t_SiN/l_SiN(col))));
    output_array_N_bg=cat(2,output_array_N_bg,double(N_bg));
```

```
    N_0=D*d_min^2/4;

    end
 end


 if (mode=="BF_TEM")  %When BF-STEM is selected
    N_0=D*d_min^2/4;
    for col=1:size(beta,2) %beta variation
       M_signal=N_0*(exp(1)^(-(d_min/l_o(col)+(t-
d_min)/l_H2O(col)+2*t_SiN/l_SiN(col))));
       M_bkg=N_0*(exp(1)^(-(t/l_H2O(col)+2*t_SiN/l_SiN(col))));
       SNR=abs(M_signal-M_bkg)/sqrt(M_bkg)==3;% Absolute value of the signal taken
as the contrast can be negative or positive
       num_sol = vpasolve(SNR, d_min); %numerical solution by using VPASOLVE;
       if isempty(num_sol)==1
          output_array_d_min=cat(2,output_array_d_min,NaN);
       end

       if isempty(num_sol)==0
          output_array_d_min=cat(2,output_array_d_min,double(num_sol));
       end
    % Six following lines use the numerically solved d_min to calculate the number of
electrons that hit the detector when the probe is located on an object  (N_o and N_bk,
respectively.)
    % This was done to double check that the negative values of d_SNR_min  obtained
from the numerical solution are indeed due to the contrast inversion.
    num_sol=abs(num_sol); %Absolute value of the numerically solved d_min.
    N_0=D*num_sol^2/4; %Number of incident electrons on anoptimized pixel
    N_o=N_0*(exp(1)^(-(num_sol/l_o(col)+(t-num_sol)/l_H2O(col)+2*t_SiN/l_SiN(col))));
    output_array_N_o=cat(2,output_array_N_o,double(N_o));
    N_bg=N_0*(exp(1)^(-(t/l_H2O(col)+2*t_SiN/l_SiN(col))));
    output_array_N_bg=cat(2,output_array_N_bg,double(N_bg));
    N_0=D*d_min^2/4; % this is needed to define the N_0 again for the numerical solver
    end
 end
```

```
  d_snr_num=output_array_d_min; %output d_SNR vs. beta
  N_o=output_array_N_o; %output for the Number of electrons that hit the detector when the probe is on object
  N_bg=output_array_N_bg; %output for the Number of electrons that hit the detector when the probe is on background



end
```

SI 3.1 Description of the Image analysis pipeline for determining the diameter of PMs

The goal of this image processing task was to segment PMS particles in the field of view and report their average radius. Due to the low SNR of PMS in liquid, the direct segmentation sequence in Step 1 registered unwanted random background fluctuation as objects. To circumvent the problem, the analysis was focused on features of which location remained constant within at least 2/3 (or 66%) during the sequence; Hence, these segmented features were characterized as having "temporal persistence". In order to first quantify the temporal persistence used for the selection of the features of interest, All objects in the image sequence were segmented then the whole stack was averaged. This effectively transformed the temporal persistence into *n* levels of gray (Step 2), where $0 \geq n \leq$ number of frames in the sequence. After this, we binarized the image by setting the threshold value to 66% resulting in a binary image including only features fulfilling the above stated criteria for temporal persistence. We performed further analysis in the area around these persistent features by centering a 200 *200 nm rectangular regions of interest (ROI) in the center of every feature. In Step 4. features within each of these ROIs were analyzed as individual images so that only features fully fitting the square were included. (Step 4) This approach effectively removed false positives from the analysis and made it easy to visualize the success of the image processing pipeline for a given dataset. (Step 6.) After analyzing the rectangular regions of interest, the script listed segmented features with total areas and their center-of mass-coordinates. The latter is defined in the ImageJ documentation as the brightness-

weighted average of x- and y-coordinates within the object . The obtained list contained some duplicates with identical area and location, and these were removed manually using Microsoft Excel-software. Next, the data was imported to the Prism v. 7.03 (GraphPad, San Diego, US) to statistically evaluate the size distribution of PMS as a function of the D_e. In brief, we used the one-way analysis of variances (ANOVA) to show that there is a statistically significant decrease in the average diameter in subsequent frames of the dataset.

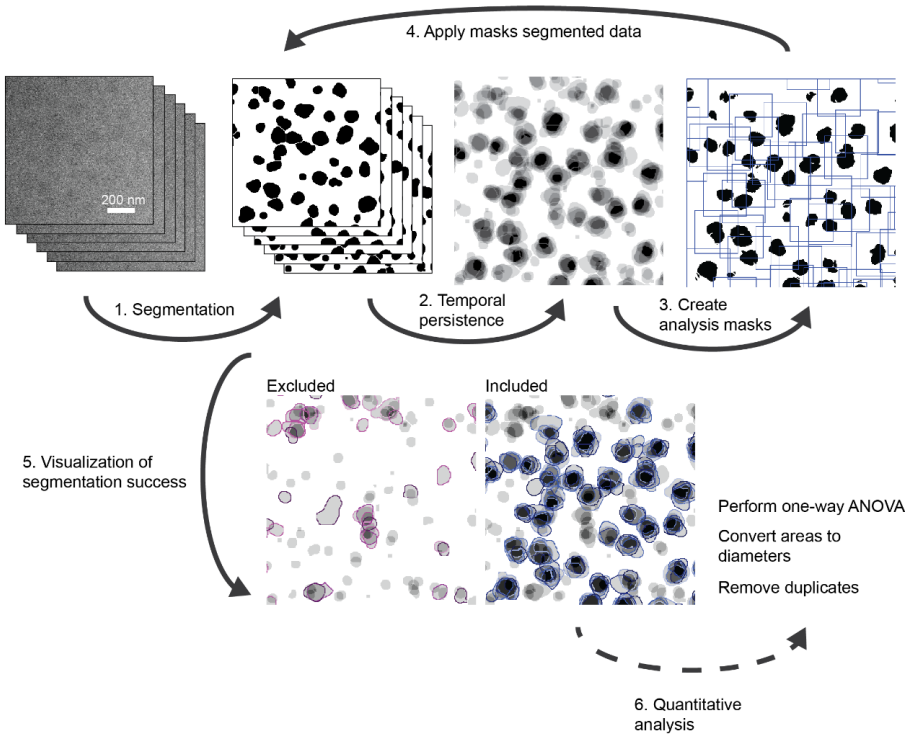

Figure S.1 The image processing pipeline used to analyze the size distribution of hydrated polystyrene microspheres (PMs) in a sequence of images. In steps 1-3, separate regions of interest were defined that were quantitatively analyzed in steps 4-6. In 5, the visualization of segmentation success shows objects that were excluded (pink) and included (blue) to the quantitative analysis done in Prism-software.

The following steps were applied to analyze an image series:

Step 1.

- Conversion to an *8-bit* image

- 2 x *Bandpass filter*ing of structures larger than 100 pixels and smaller than 4 pixels
- Binarization of the image by *Auto threshold*ing using the Li-method [2]
- *Open-* a procedure to remove small specks: 8 Iterations, 3 Counts
- *Dilate-* a procedure to restore the average size of PMS to 110 nm in the first frame of the sequence: 2 Iterations, 1 Count.
- *Watershed*-operation to separate connected particles.

Step 2. Temporal Persistence of features

- All the binarized images in the stack are averaged by intensity.
- Average image *threshold is* set to 66% (2/3)
- Temporally persistent features were selected by the *Analyze particles*-tool. Only features > 250 nm^2 were included.

Step 3. Creation of masks

- 200 x 200 nm rectangular ROIs are created and centered at the center-off-mass of each temporally persistent feature.

Step 4. Application of mask to the segmented data

- For each mask, *Analyze particles*, including only features that are not touching the edges and have size > 250 nm^2 and circularity 0.5-1.0.

Step 5. Visualization of segmentation success

- Included and excluded features are drawn to help the evaluation of segmentation success.

Step 6. Quantitative analysis of size distributions

- Data including Center-of-masses and areas are exported to Microsoft Excel, where duplicates are removed and areas *A* are transformed to diameters *d* using the formula:

$$d = 2\sqrt{\frac{A}{\pi}}$$

- Data is exported to Prism GraphPad software where the one-way ANOVA is applied to determine the statistically significant changes in the size distribution.

SI 3.2 ImageJ script for analysing the PM size evolution

Input:

A stack of (aligned) images (eg. “stackOfImages.tif”) and a rectangular selection of the area to be analyzed

Output:

stackOfImages _selected area.zip

ROI of for the area that was manually selected for analysis.

stackOfImages_blue_objects.tif

Segmented objects indicated with blue out lines over an averaged stack of thresholded images .

stackOfImages _pink_background.tif

Segmented objects that were left out of the analysis due to low “temporal persistense” are indicated with pink out lines over an averaged stack of thresholded images .

stackOfImages_ Bandpass filtered.tif

Stack of bandpass-filtered images

stackOfImages_ Norm_BPF_with_objects.tif

Segmented objects indicated with blue out lines over a Bandpass-filtered stack of images.

stackOfImages _segmented blobs.txt

Measured area, center of masses (XM, XY) for each segmented object.

stackOfImages _segmented blobs.zip

ROIs for the segmented objects

```
//PRE-PROCESSING STEPS

//All other windows are closed

close("\\Others")

//Get path and filename in order to define the base of the file name

path=getDirectory("image");

file_name=getInfo("image.filename")

file_name_base=substring(file_name, 0, lengthOf(file_name)-4);


//The manually selected area for analysis is saved as ROI

roiManager("add")

roiManager("select", 0)

roiManager("rename", file_name_base+"_selected_area_")

roiManager("save", path+file_name_base+"_selected area.zip")


//STEP 1

//Here basic image processing steps are executed in order to create binary (Black&white) image of individual frames

run("Duplicate...", "duplicate");

rename("Bandpass filtered");

run("8-bit");

run("Bandpass Filter...", "filter_large=100 filter_small=4 suppress=None tolerance=5 autoscale saturate process");

run("Bandpass Filter...", "filter_large=100 filter_small=4 suppress=None tolerance=5 autoscale saturate process");

run("Duplicate...", "duplicate");

BPF_file_name=file_name_base+"_"+getInfo("window.title")

saveAs(path+BPF_file_name)

print ("Saved as "+ path+ BPF_file_name)

run("Auto Threshold", "method=Li white stack");

run("Options...", "iterations=8 count=3 do=Open stack");

run("Options...", "iterations=2 count=1 do=Dilate stack");

run("Watershed", "stack");
```

//Image stack "Segmented" is the segmented data to be analyzed after ROIs are selected

```
rename("Segmented")

run("Duplicate...", "duplicate")

segmented_file_name=file_name_base+"_"+getInfo("window.title")

saveAs(path+segmented_file_name)

print ("Saved as "+ path+ segmented_file_name)
```

//Image stack "Background" will be used to show all axcluded blobs in the analysis

```
rename("Background")
```

//STEP 2

// Here Seeds are created for centering rectangular ROIS. Averaging is used to convert temporal events into intensity variation. If there is a blob on the same pixel all the time, it will be 255 ie.black

```
run("Z Project...", "projection=[Average Intensity]")

rename("Seeds")
```

//STEP 3

//This thresholding step decides the temporal persistence of a seed and should be decided before analysis. That is, for how many time point blob needs to be on a certain ROI. For example, 75% that would correspond for a blob being on a same spot for 3/4 of the sequence.

```
setThreshold(0, 111);

setOption("BlackBackground", false);

run("Convert to Mask");

run("Analyze Particles...", "size=250-Infinity pixel circularity=0-1.00 display exclude clear in_situ");

seeds_file_name=file_name_base+"_"+getInfo("window.title")

saveAs(path+seeds_file_name)

print ("Saved as "+ path+ seeds_file_name)
```

//size of the rectangle that is used as ROI at the coordinates of each seed.

d=65

//This counter is used to separate rectangle rois at the beginning of the Roimanager from actual objects

```
counter=0
```

```
//removes selection
makeRectangle(1, 1, 1, 1);
roiManager("add")
roiManager("Deselect");
roiManager("Delete");

//Extracts center of masses of seeds from the results table and draws an rectangular ROI (define size, d )
for (i=0; i<nResults; i++) {
        x_co=getResult("XM",i);
        y_co=getResult("YM",i);
        makeRectangle(x_co-d/2, y_co-d/2, d, d);
        roiManager("add");
        counter=counter+1;
        print(counter);
        }

//Every rectangular ROI is saved them to the same folder as a .zip file
roiManager("save", path+file_name_base+"Analyzed ROIs.zip");

//STEP 4
//Analysis of segmented data based on the Analysis ROIs created in Step 3.
selectWindow("Segmented")
run("Invert", "stack");
run("Clear Results");

for (j=1 ; j<(nSlices()+1); j++) {
        print(nSlices);
        for (i=0;i<counter;i++){
                roiManager("select", i);
                setSlice(j);
                run("Analyze Particles...", "size=250-Infinity pixel circularity=0.50-1.00 display exclude add in_situ");
                print(counter);
```

```
        }
}
run("Invert", "stack");

//Saves analysis as _segmented_blobs.txt.
selectWindow("Results")
saveAs("results", path+file_name_base+"_segmented blobs.txt")

//Rectangular rois are deleted
roiManager("select", 0)
for (i=0 ; i<counter; i++) {
   roiManager("select",0);
   roiManager("delete");
}

//freeline Rois are  saved into a zip file segmented blobs.zip
roiManager("save", path+file_name_base+"_segmented blobs.zip");

//STEP 5
// Only  objects excluded from the analysis are shown in "background"image and their outlines are drawn in pink.
run("Clear Results");
selectWindow("Background");
setForegroundColor(255, 255, 255);
for (i=0;i<(roiManager("count")); i++) {
        roiManager("select",i);
        roiManager("Fill");
}
background_file_name=file_name_base+"_"+getInfo("window.title")
file_name_base=substring(file_name, 0, lengthOf(file_name)-4);
saveAs(path+background_file_name)
print ("Saved as "+ path+ background_file_name)

roiManager("deselect");
```

```
roiManager("delete");


selectWindow("Background");
for (j=1 ; j<(nSlices()+1); j++) {
	print(nSlices);
	setSlice(j);
	run("Analyze Particles...", "size=250-Infinity pixel circularity=0.5-1.00 display exclude
add in_situ");
}


selectWindow("Results")
saveAs("results", path+file_name_base+"_background blobs.txt")
```

//ROI's of excluded objects are saved as zip

```
roiManager("save", path+file_name_base+"_background_blobs.zip");


roiManager("deselect");
```

//Draw pink outlines

```
roiManager("Show None");
roiManager("Show All");
selectWindow("Background");
selectWindow("Background")
roiManager("deselect");
run("Duplicate...", "duplicate");
rename("Pink_background");
run("RGB Color");
run("Z Project...", "projection=[Average Intensity]")
rename("AVG_pink_background");



for (i=0;i<roiManager("count");i++){
	selectWindow("Pink_background");
	roiManager("select", i);
```

```
	slice=getSliceNumber();
	print(slice);
	if (slice==1) {
		setForegroundColor(249, 185, 255);
	}
	if (slice==2) {
		setForegroundColor(245, 139, 255);
	}
	if (slice==3) {
		setForegroundColor(241, 39, 255);
	}
	if (slice==4) {
		setForegroundColor(255, 000, 255);
	}
	if (slice==5) {
		setForegroundColor(169, 000, 185);
	}

	if (slice>5) {
		setForegroundColor(85, 000, 93);
	}
	selectWindow("AVG_pink_background");
	roiManager("draw");
	selectWindow("Pink_background");
	roiManager("draw");
}

selectWindow("AVG_pink_background");
AVG_pink_file_name=file_name_base+"_"+getInfo("window.title")
file_name_base=substring(file_name, 0, lengthOf(file_name)-4);
saveAs(path+AVG_pink_file_name)
print ("Saved as "+ path+AVG_pink_file_name)
roiManager("deselect");
```

```
roiManager("delete")
roiManager("open", path+file_name_base+"_segmented blobs.zip");
selectWindow("Segmented");
roiManager("deselect");
selectWindow("Segmented");
```

//// Only objects included from the analysis are shown in "Segmented"-image and their outlines are drawn in blue.

```
roiManager("Show None");
roiManager("Show All");
selectWindow("Segmented");
roiManager("deselect");
run("Duplicate...", "duplicate");
rename("Blue_objects");
run("RGB Color");
run("Z Project...", "projection=[Average Intensity]")
rename("AVG_blue_objects");
selectWindow("Bandpass filtered");
run("Duplicate...", "duplicate");
run("Enhance Contrast...", "saturated=0.3 normalize process_all");
rename("Norm_BPF_with_objects");

for (i=0;i<roiManager("count");i++){
        selectWindow("Norm_BPF_with_objects");
        roiManager("select", i);
        slice=getSliceNumber();
        print(slice);
        if (slice==1) {
                setForegroundColor(185, 200, 255);
        }
        if (slice==2) {
                setForegroundColor(139, 164, 255);
        }
```

```
		if (slice==3) {
			setForegroundColor(93, 127, 255);
		}
		if (slice==4) {
			setForegroundColor(000, 054, 255);
		}
		if (slice==5) {
			setForegroundColor(000, 030, 139);
		}

		if (slice>5) {
			setForegroundColor(000, 20, 93);
		}
		selectWindow("AVG_blue_objects");
		roiManager("draw");
		selectWindow("Norm_BPF_with_objects");
		roiManager("draw");
}

selectWindow("AVG_blue_objects");
AVG_blue_file_name=file_name_base+"_"+getInfo("window.title")
file_name_base=substring(file_name, 0, lengthOf(file_name)-4);
saveAs(path+AVG_blue_file_name)
print ("Saved as "+ path+AVG_blue_file_name)

selectWindow("Norm_BPF_with_objects");
Norm_BPF_with_objects_file_name=file_name_base+"_"+getInfo("window.title")
file_name_base=substring(file_name, 0, lengthOf(file_name)-4);
saveAs(path+Norm_BPF_with_objects_file_name)
print ("Saved as "+ path+Norm_BPF_with_objects_file_name)

run("Tile");
```