17. Annexes

17.1 Crystal Field Theory

Crystal field theory is a simplified approach to bond geometry in complexes and solids that only considers electrostatic interactions between ligands and the coordinated atom, treating them as point charges, to explain the orbital energy values. The main explanatory consequence of this model is the splitting of some valence orbitals (usually d orbitals) energy values, due to the (a)symmetry of arrangement of the ligands. [124]

Because an example is often the best way to understand a concept, the next paragraph illustrates this theory in practice. The following example is illustrated on Figure 98.

If some number of negative charges were spread in a sphere around a metal cation, the cation d orbitals would understandably be destabilized, as negative charges repel each other and any electrons in these orbitals would gain potential energy.

However, ligands do not spread their negative charge uniformly around the cation. For an octahedral bond geometry, the ligands are all located along the x, y and z axes. As a result, the arrangement of the ligands directly faces the $d_{x^2-y^2}$ and the d_{z^2} orbitals. Any electrons in these orbitals (now called e_g orbitals) would see their energy increased via electrostatic repulsion with the ligand electrons. On the other hand, the d_{xy}, d_{xz} and d_{yz} orbitals (now called the t_{2g} orbitals) are facing away from the ligands, thus their energy is reduced relative to e_g (they are still destabilized by the ligand repulsive charges, but less than e_g). Since the d orbitals are now divided into e_g and t_{2g} orbitals of different energy, they are said to have undergone a degeneracy lift. [125,126]

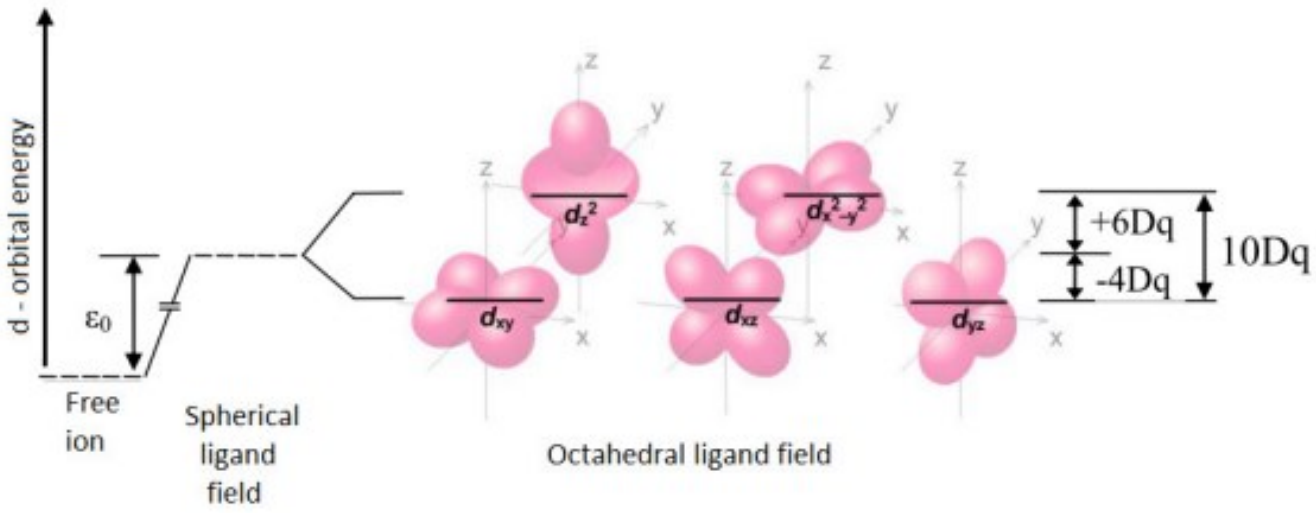


Figure 98 Graphical explanation of CFT [20]

The resulting orbitals will be different according to the ligand arrangement. Other common arrangements such as square-planar, tetrahedral, cubic all differ by the resulting orbitals, but they all result in some degree of degeneracy lift since the spherical symmetry is broken.

In any case, CFT provides an expression for the magnitude of the degeneracy lift, called crystal field splitting:

$$\Delta_o = \frac{5}{3}\frac{q}{R^5}\langle r^4 \rangle$$

(1)

With q = ligand charge, R = metal-ligand distance, $\langle r^4 \rangle$ = mean value of the fourth power of the d-electron-nucleus distance. [125]

However, that expression rarely accurately predicts this value, the reason for which will be explained in the next section. It should be noted that the crystal field splitting (Δ_o on Figure 98) between e_g and t_{2g} orbitals is related to the magnitude of the electrostatic interactions (but not only) between metal and ligand (higher electrostatic interaction raises the e_g energy level) and is important for the spin transition phenomenon. The limits of this theory are shown in the spectrochemical series. This series ranks some ligands from the weakest to the strongest ligand field Δ_o:

$$I^- < Br^- < SCN^- < S^{2-} < Cl^- < NO_3^- < F^- < OH^- \sim RCOO^- < H_2O \sim RS^- < NH_3 \sim \text{Im (imidazole)} < \text{bpy (2,2'-bipyridine)} < CN^- < CO$$

Indeed, crystal field theory fails to explain how H_2O produces a stronger field splitting than OH^-, which due to its negative charge should exert a stronger electrostatic force on the metal centre. [124]

17.2 Ligand Field Theory

As demonstrated with the spectrochemical series, the field splitting value Δ_o is not properly explained/predicted by CFT. This is because electrostatic effects are not the only effects to consider, and ligands can hardly be considered as point charges. This is the reason why ligand field theory was introduced. [20] This theory uses molecular orbital theory to explain these discrepancies. The 3d orbitals once again represented on Figure 98 are helpful to understand this principle. For an octahedral coordination geometry, assuming that every ligand possesses one σ orbital, we can see that the $d_{x^2-y^2}$ and d_{z^2} (along with the s, p_x, p_y, p_z) orbitals lie in the direction of the σ metal-ligand bond, hence are able to produce σ bonds (bonding or anti-bonding). On the other hand, the orientation of the d_{xy}, d_{xz} and d_{yz} orbitals renders σ bonding impossible, and only π bonding is possible. [127]

The resulting molecular orbital (MO) diagram for an octahedral complex, produced on Figure 100, reveals a similar result to that of CFT, with the 3d orbital being split into e_g and t_{2g} orbitals (with the exception that with LFT, e_g is identified as an antibonding orbital e_g^*). As such, the conclusions of CFT and LFT are not that different in terms of orbitals. The strength of LFT is that it explains how the field splitting must be determined not only by electrostatic interaction but also by modifying the t_{2g} orbital energy according to the energy level of the ligand π orbitals, as seen on Figure 99. [124,128]

If the ligand has filled π orbitals (left situation on Figure 99, π -DONOR), the interaction with the d_{xy}, d_{xz} and d_{yz} orbitals produces bonding orbitals which are filled by the ligand π electrons and anti-bonding orbitals of lower energy than e_g*, which become the t_{2g} orbitals. The resulting Δ is lower with this π interaction. If the ligand has empty π orbitals (right situation on Figure 99, π -ACCEPTOR), the interaction with the metal t_{2g} orbitals will produce anti-bonding orbitals of higher energy than e_g* and thus will not be used. It will also produce bonding orbitals which become the new t_{2g} orbitals, resulting in a bigger Δ. [124]

In conclusion, to properly determine the orbitals energy levels, both crystal- and ligand field theories should be considered, as both electrostatic and molecular effects influence the field splitting.

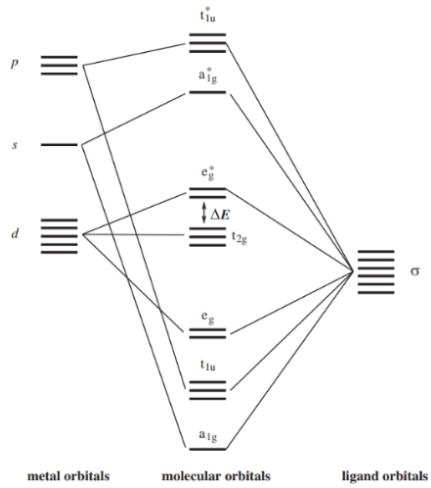


Figure 100 Molecular orbitals of an octahedral complex, e_g and t_{2g} orbitals are seen in the middle of the diagram* [18]

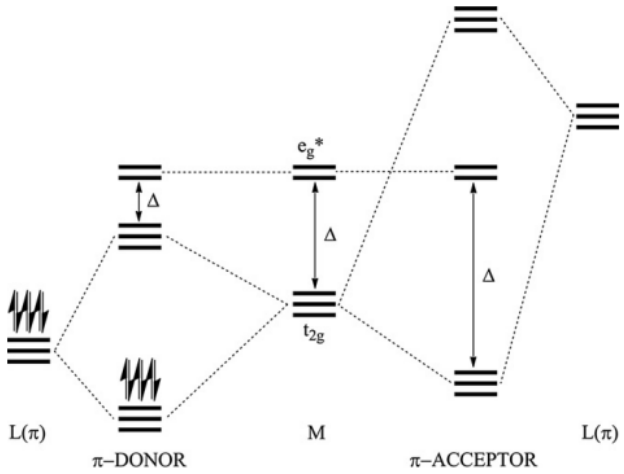


Figure 99 Graphical illustration of the variation of Δ through the ligand field theory [18]

17.3 State of the Art : Fe^{III},Co,Mn,Ni

Fe^{III} compounds can also be found in mononuclear, dinuclear, or polymer form, but trinuclear or higher are not known for Fe^{III}, and only 1D polymers have been currently synthesized (no 2, 3D networks yet). The advantage of these materials is their stable oxidation state which remove difficulties in synthesis. Their behaviour is very similar to Fe^{II}

complexes, and their possible coordination spheres are very varied. N_4O_2, $N_2O_2S_2$, N_3O_2S and N_5O are all coordination spheres which have been found to induce SCO. There is untapped potential in these compounds, since over the last 15 years, more than 12 SCO compounds which have a hysteresis greater than 10K have been found, but no compound with large hysteresis at room temperature has been found still. [42]

Cr^{2+} complexes showing spin-switching properties are very few, with only three types having been discovered (Figure 101). The proposed reason for this scarcity is that, as a d^4 ion, the spin transition from LS to HS results in a smaller entropy gain (as discussed previously, spin crossover is mostly entropy driven), thus a spin crossover compound is more difficult to obtain. However, Mn^{3+} also is a d^4 system and it has produced much more active complexes. For this reason, it has been proposed that the difficulties of Cr^{2+} compounds are due to the high oxidation sensitivity of $Cr^{2+} \rightarrow Cr^{3+}$, which is a d^3 ion which therefore cannot switch spin states. [43,44]

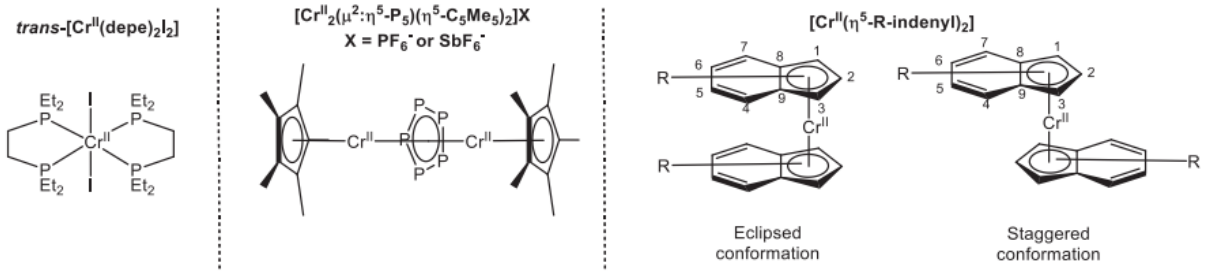


Figure 101 - 3 types of Cr2+ SCO complexes [43]

Of the seven oxidation state of manganese, only 2 have presented a SCO activity, namely Mn^{2+} and Mn^{3+}. Both generate octahedral complexes capable of SCO, Mn^{2+} switching from S = 5/2 HS to S = ½ LS state, while Mn^{3+} switches from S = 2 HS to S = 1 LS states. Again, Mn^{3+} compounds are far more numerous due to the worse oxidation stability of Mn^{2+}. Indeed, up to now, only 2 Mn^{2+} SCO complexes have been identified, and none have had hysteretic behaviour.[129,130]

Mn^{III} compounds are usually composed of a N_4O_2 coordination sphere. While a multitude of SCO compounds can be found in the literature, only 4 hysteretic compounds have been found so far, all centred at low temperatures. Namely, [MnL1]PF6 (Figure 102), [Mn(sal-N-1,5,8,12)]SbF_6, [Mn(5-Cl-sal-N1,5,8,12)]$TCNQ_{1.5}\cdot 2CH_3CN$, [Mn(sal-N-1,5,8,12)]AsF_6. However, for the former two compounds, the hysteresis is not brought by a classic spin transition, but rather a change in the H bond network induced by an anion (PF_6, SbF_6) phase change. [43,129,131,132]

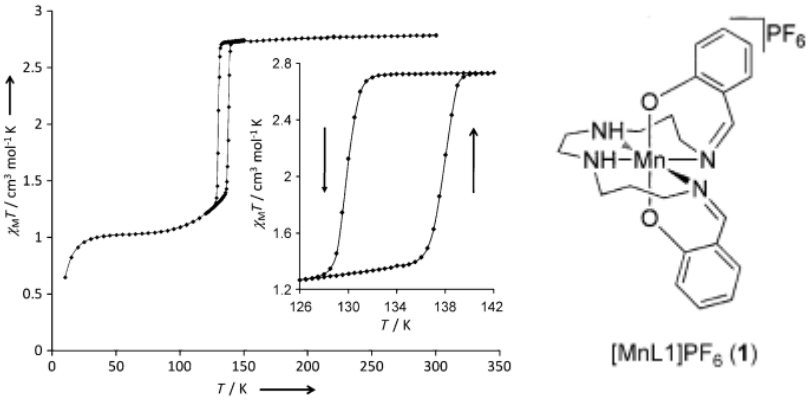


Figure 102 The first example of an abrupt hysteresis for a Mn^{III} compound. [26]

In the field of cobalt SCO research, obtaining abrupt hysteretic spin transitions has proven to be a challenge. Also, not all cobalt compounds exhibit thermochromic properties, which limits their applicability in emerging technologies.

While being much less common than Fe^{II} complexes, a few examples do show hysteretic properties, though usually at low temperatures, such as the mononuclear $[Co^{II}\text{-}(dpzca)_2]$ for example with a $T_{1/2}{\uparrow}$ = 179K and $T_{1/2}{\downarrow}$ = 168K (ΔT = 11K). [133] Other examples include $[Co^{II}(pyterpy)_2](PF_6)_2{\cdot}2MeOH$ ($T_{1/2}{\uparrow}$ = 191 K and $T_{1/2}{\downarrow}$ = 182 K, ΔT = 9K) [134]. As seen through these example, Co^{II} SCO complexes are commonly made up of a N_6 coordination sphere, like most Fe^{II} complexes. The particularity of Co^{II} is that, in order to obtain a ligand field splitting that permits the switch from both states, these 6 coordinated nitrogen are part of heterocycles. Both examples above show the use of terpyridine-like ligands, which would produce a too high field splitting in Fe(II) complexes. [43]

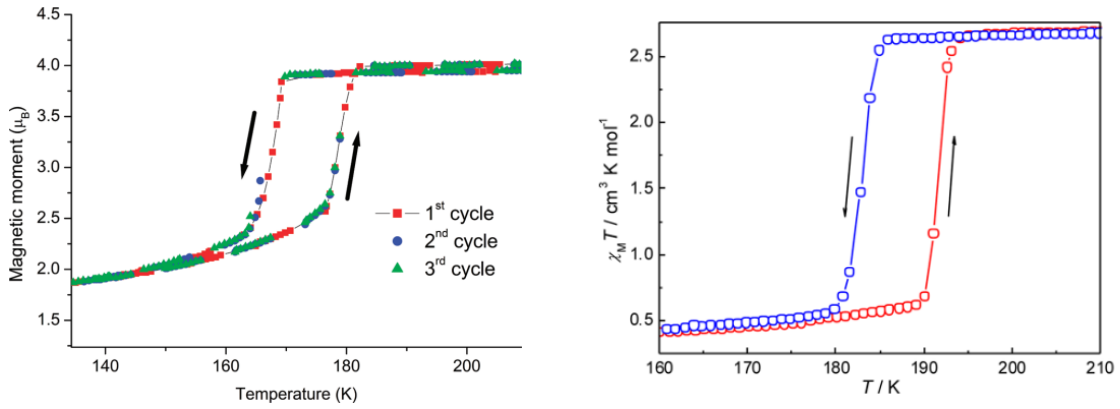


Figure 103 Right : XT versus T plot of $[Co^{II}(pyterpy)_2](PF_6)_2{\cdot}2MeOH$. [31] *Left : XT versus T plot of $[Co^{II}\text{-}(dpzca)_2]$*

Recently, researchers have managed to obtain room temperature hysteresis Co^{II} SCO compounds, by using ligands with long alkyl chains, in order to improve intermolecular interactions, and thus cooperativity. [135]

Another possibility that this strategy brings about is a temperature induced phase transition of the alkyl chains, triggering what is called a reverse spin transition. This is the case for $[Co(C14\text{-}terpy)_2](BF_4)_2$ (C14-terpy = 4'-tetradecyloxy-2,2':6',2''-terpyridine), with a drop from HS to LS at $T_{1/2}\uparrow$ = 307K and a climb at $T_{1/2}\downarrow$ = 250 K, as seen on Figure 104. [40,136]

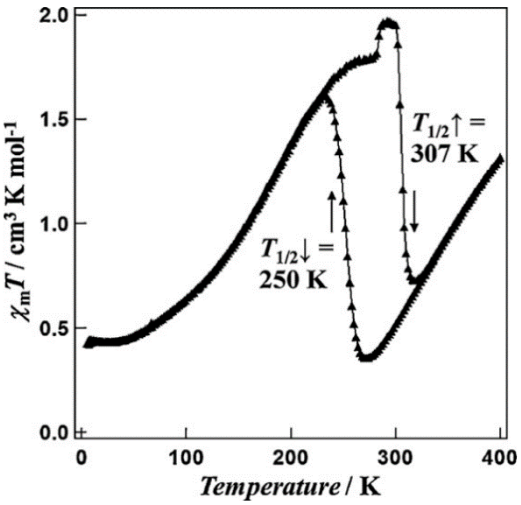
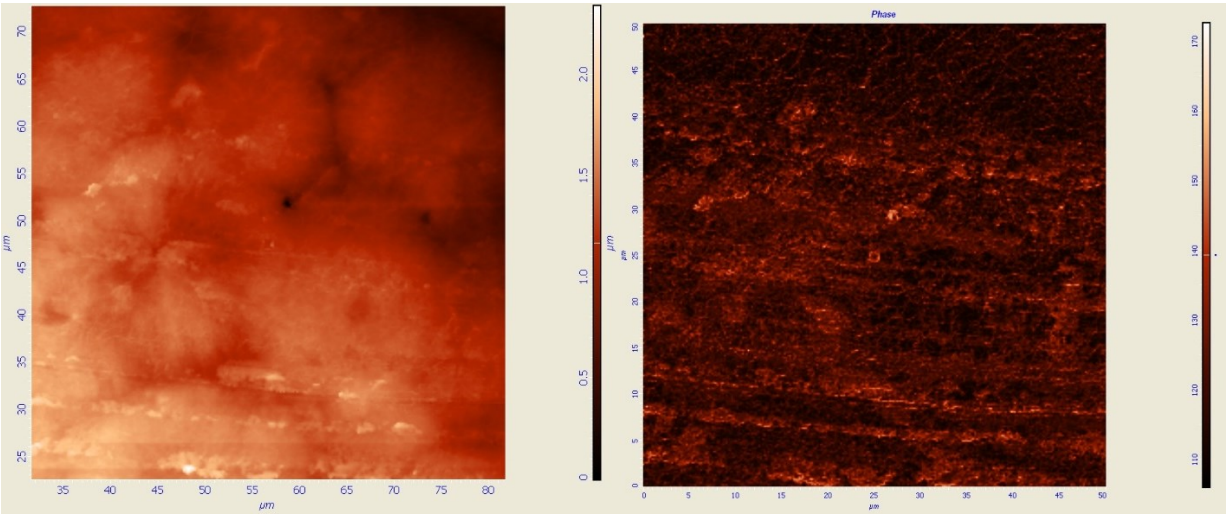
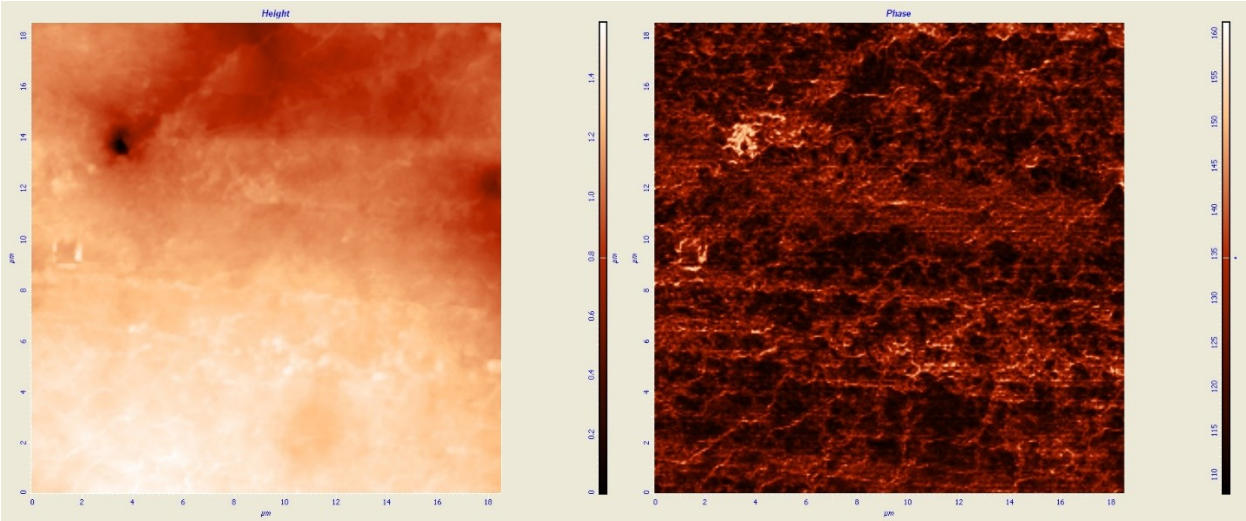


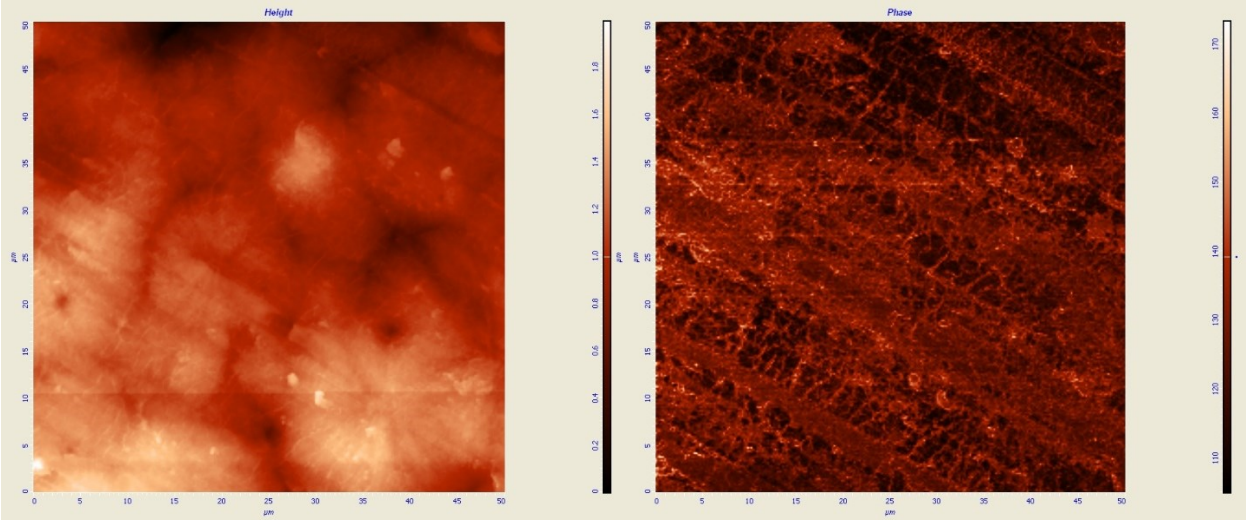
Figure 104 Reverse spin transition in the Co compound $[Co(C14\text{-}terpy)_2](BF_4)_2$ [125]

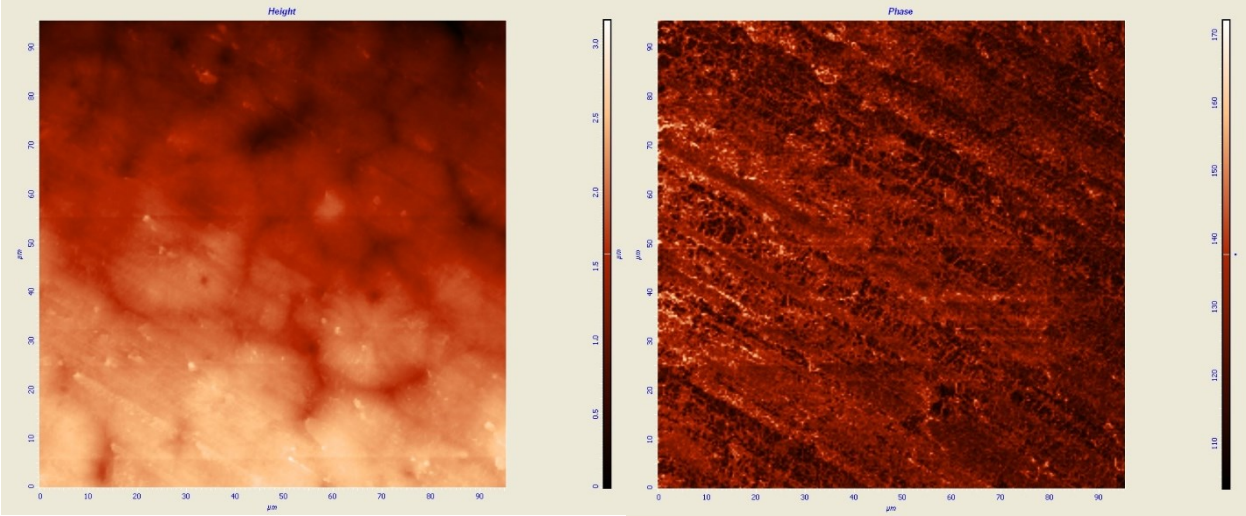
It is worth mentioning that Ni^{II} complexes can present SCO-like characteristics, with a transition from HS to LS not due to a spin transition per se, but rather a change of complexation geometry from tetrahedral (HS) to square-planar (LS). It is thus not generally considered a real spin transition. [44,137]

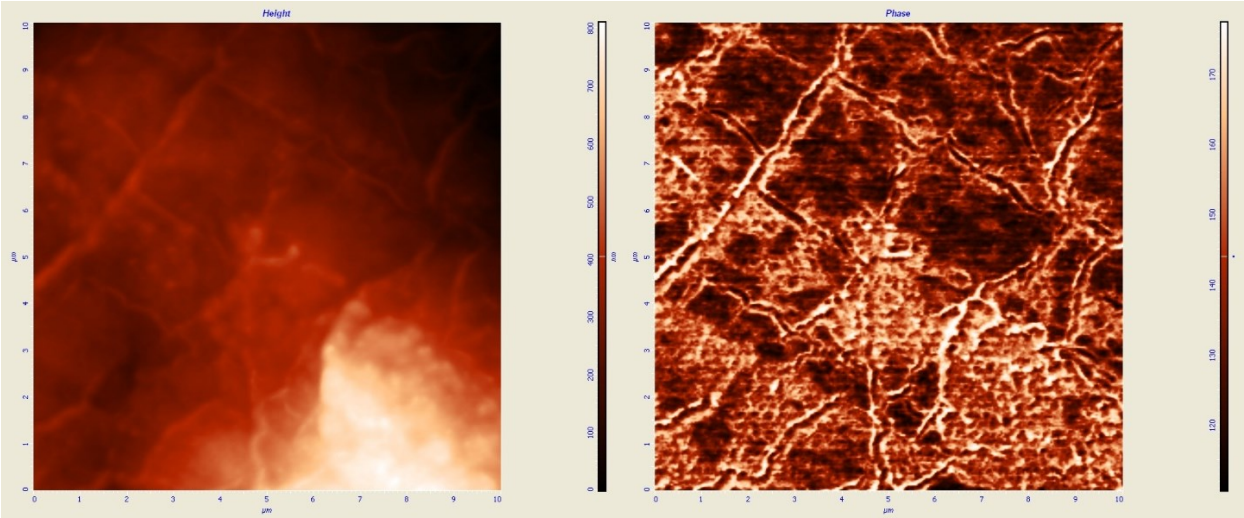
17.4 Polymer AFM images (Height = left, Phase = right)

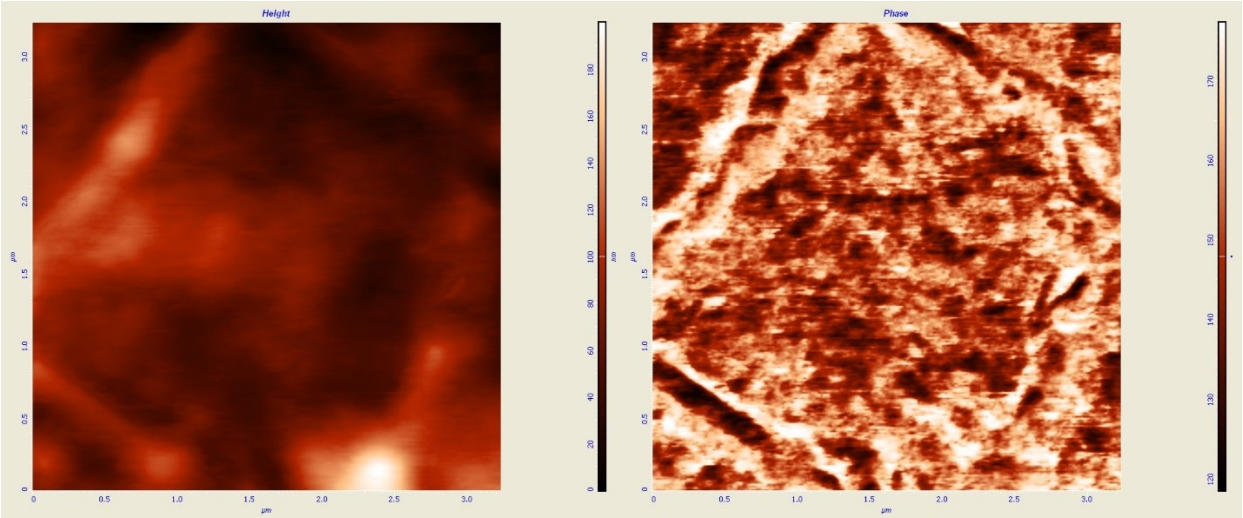


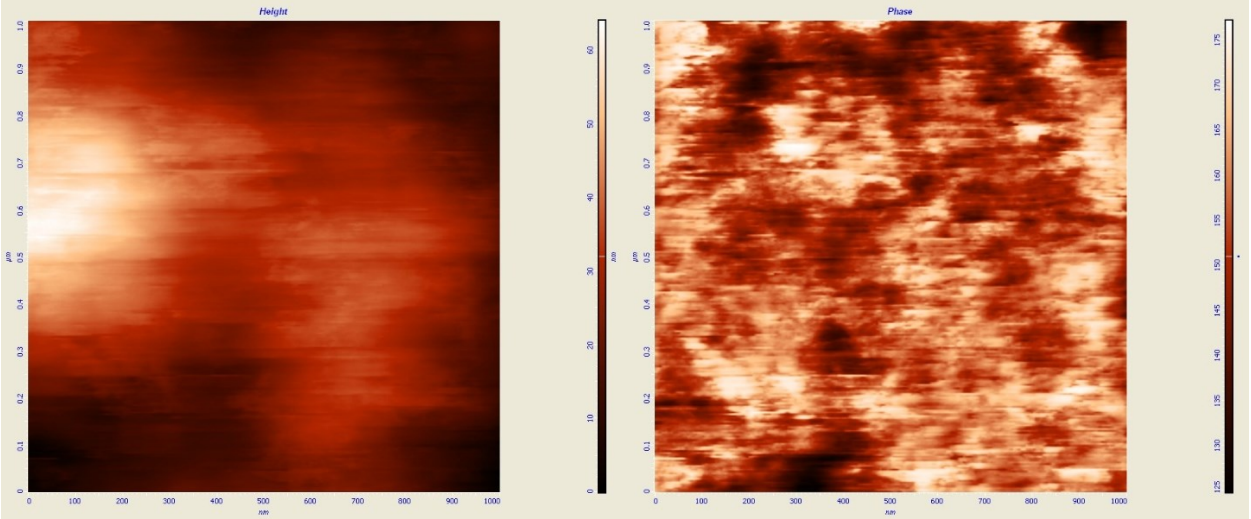

Height
Phase

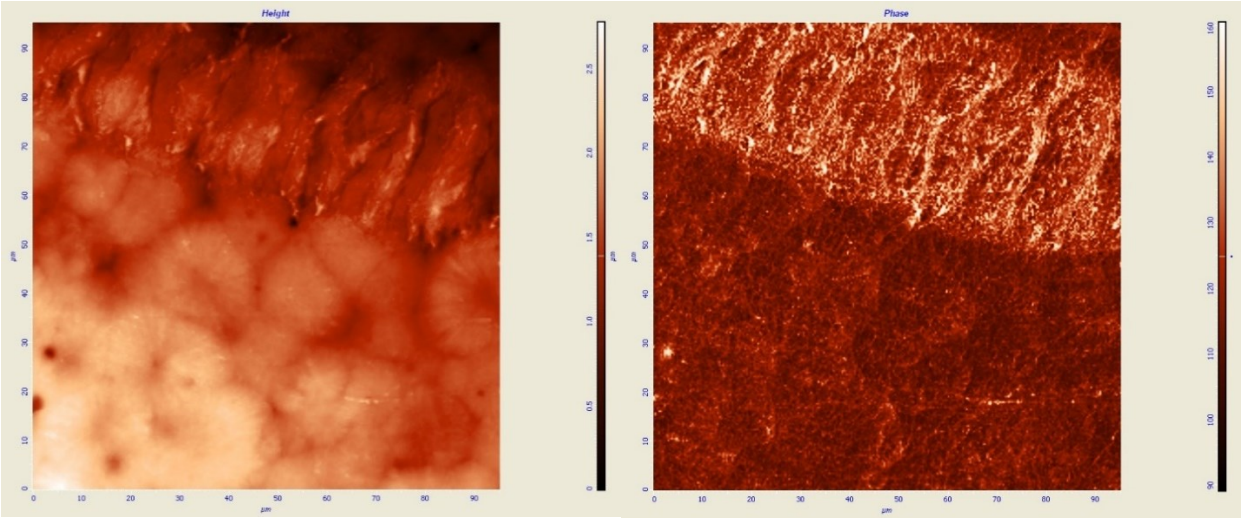

Height
Phase


Height
Phase

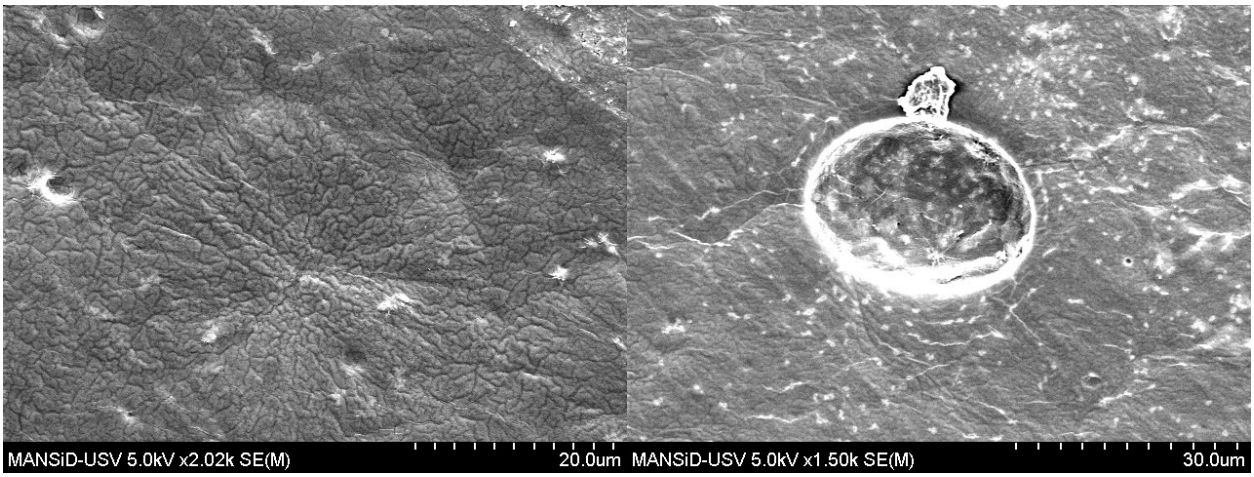

Height
Phase

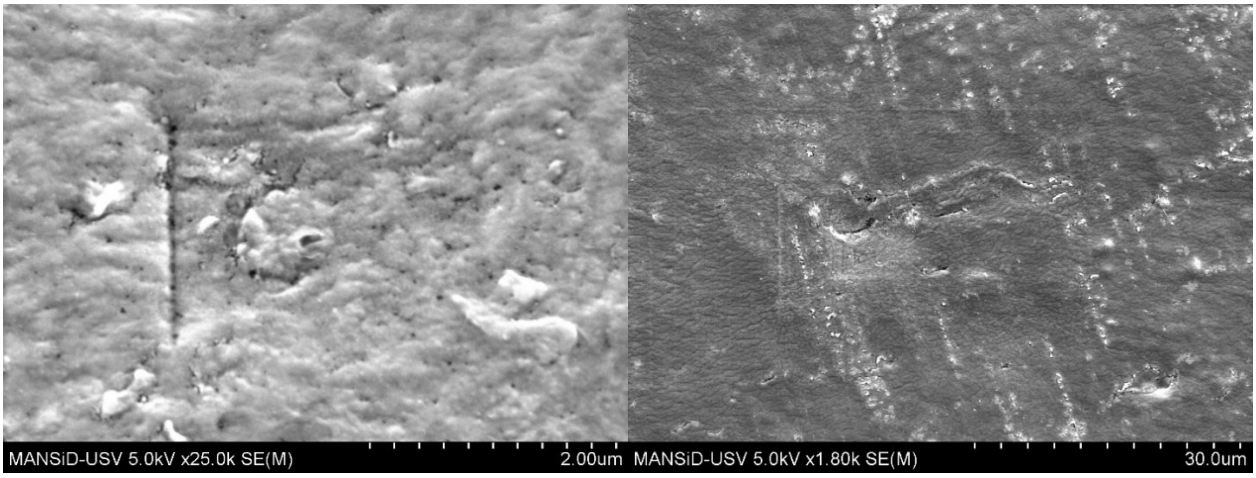

Height
Phase

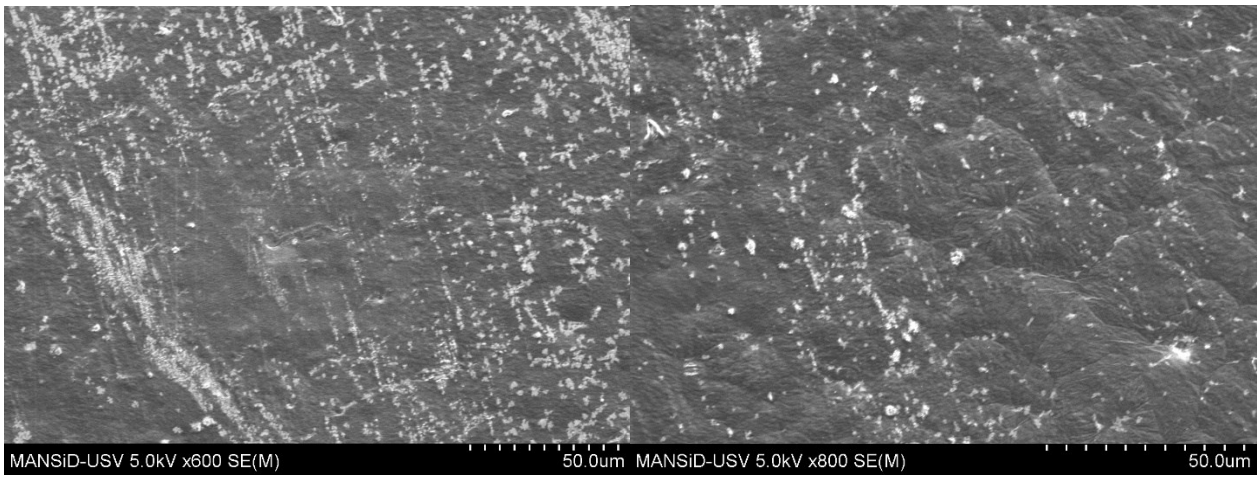

Height
Phase

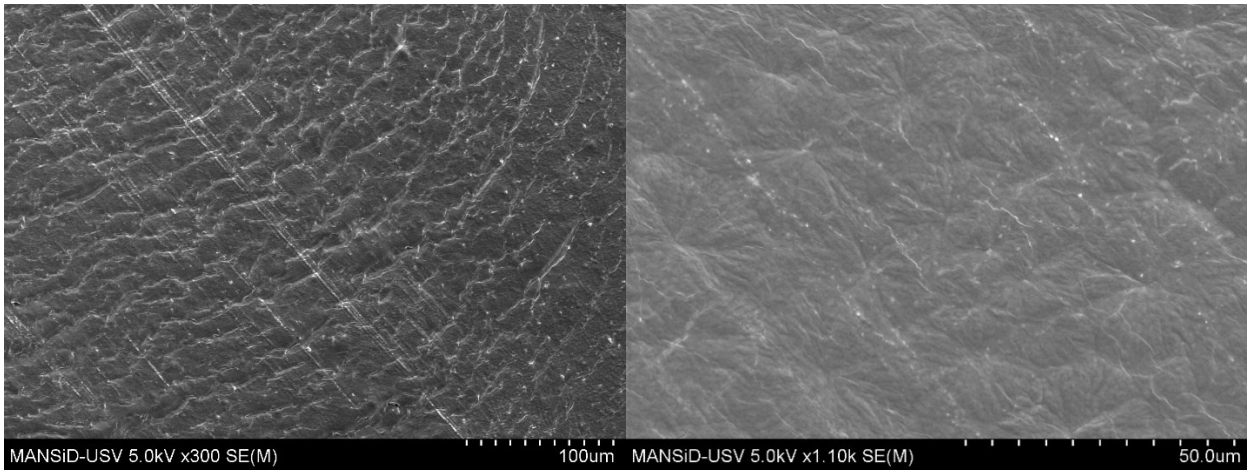


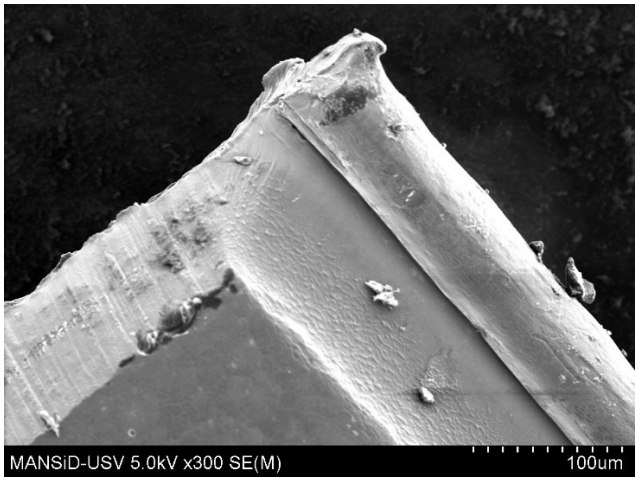
17.5 Polymer SEM images











17.6 [Fe(Htrz)3-3x(NH2trz)3x](ClO4)2·nH2O

The idea of using ligand alloys to synthesize novel triazole compounds is not new, with $[Fe(Htrz)_{3-3x}(NH_2trz)_{3x}](ClO_4)_2 \cdot nH_2O$ being first published in 1993. [138] As can be seen on Figure 105, the compound does present hysteresis at room temperature. The compound was

successfully synthesized, however multiple reasons led to the decision of discarding it for the remainder of this work :

- Firstly, the perchlorate counter anion involved is unstable and can actually produce an explosion at high temperatures, pressures, or due to friction forces. As a surface coating, the compound could be subjected to different stimuli that might trigger the explosion. [93]
- Secondly, the compound exclusively presents the desired hysteretic properties when it is hydrated. However, the solvent molecules have a high tendency to leave the compound, even at room temperature (> 25 °C).

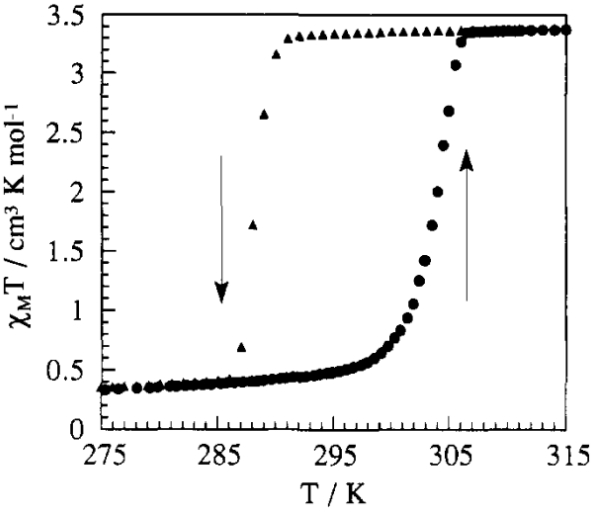


Figure 105 Magnetic properties of $[Fe(Htrz)_{3-3x}(NH_2trz)_{3x}].(ClO_4)_2 \cdot nH_2O$ as a function of temperature (x = 0.05)

17.7 [Fe1-xZnx(NH2trz)3]Cl2

$[Fe(NH_2trz)_3]Cl_2$ is a typical aminotriazole SCO coordination chain with, as a reminder, the Cl^- anions placed between the chain, constituted of Fe^{II} metal centres each sharing 6 aminotriazole ligands. This compound presents a hysteresis effect above room temperature (see Figure 106), so it must be tailored to be optimal for our purposes. Although, it can be noticed on Figure 106 that the hydrated form of this compound ($[Fe(NH_2trz)_3]Cl_2 \cdot 1{,}5H_2O$) reaches room temperature bistability, but with a too narrow hysteresis width to be practical. The influence of solvent in the polymer lattice is typical of SCO compounds and has been mentioned previously in a previous section. [81]

The strategy employed to try to obtain a compound with room temperature hysteresis was to dilute the Fe^{II} centre with Zn^{II}, thereby reducing cooperativity and bringing the transition temperatures of the compound down. Two formulations where tested : $\mathbf{[Fe_{0.95}Zn_{0.05}(NH_2trz)_3]Cl_2}$ and $\mathbf{[Fe_{0.9}Zn_{0.1}(NH_2trz)_3]Cl_2}$, which can be thought of as **random copolymers** of both the $Fe(NH_2trz)_3^{2+}$ and $Zn(NH_2trz)_3^{2+}$ subunits (or monomers). However,

both formulations returned a purple compound (LS) without hysteresis at room temperature. This means that further dilution would be needed to further approach room temperature. However, the idea was discarded because as cooperativity is lost, the hysteresis becomes narrower until a point when the compound switches to a regular spin transition rather than a hysteretic behaviour (which is required for information storage). Although this attempt has failed, it was mentioned because Zn^{II} centres will show their usefulness later in this work.

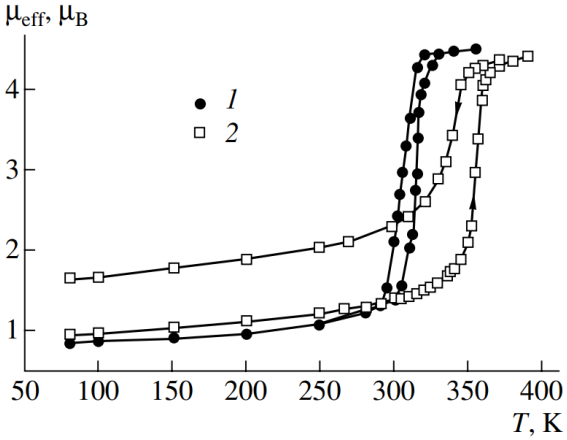


Figure 106 Magnetic properties of $[Fe(NH_2trz)_3]Cl_2$ as a function of the temperature for 1) the hydrated compound 2) non hydrated compound. [81]

17.8 [Fe(NH2trz)3](NO3)2-x(BF4)x

$[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x$ was first mentioned in 1998 by O.Kahn and C.Martinez. It is an anionic alloy of NO_3^- and BF_4^-. For x = 0.3, it presents extremely interesting properties with a very large hysteresis (50 K) centred around room temperature (see Figure 107). [139]

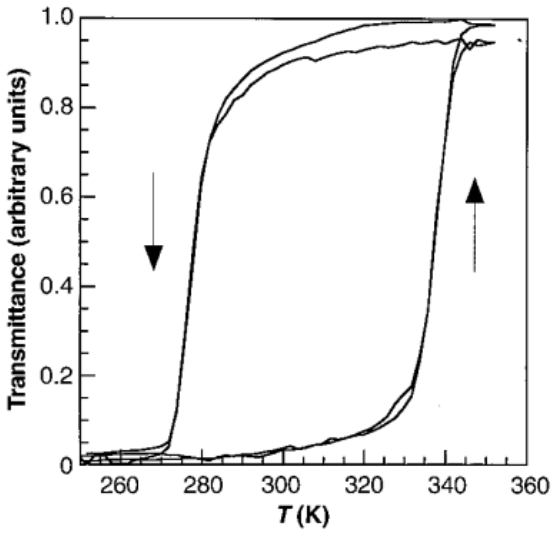


Figure 107 Reported magnetic properties of $[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x$, x = 0.3 [139]

Unfortunately, there has been no publication about the synthesis pathway for this compound. Thus, to obtain the right compound it was necessary to draw up our own synthesis protocol. The main challenge about this protocol is finding the right iron salts ratio ($[Fe(H_2O)_6](NO_3)_2$ and $[Fe(H_2O)_6](BF_4)_2$) to introduce in the reaction. Indeed, for anionic alloys, the final compound formulation will generally not correspond to the quantities introduced. As a consequence, finding the right protocol is not as straightforward as simply introducing 15% of $Fe(BF_4)_2$ and 85% $Fe(NO_3)_2$. Another difficulty that must be accounted for is that, once again, the spin transition properties of this compound are different when it is hydrated. This is of importance since, in the polyurethane coating , the compound loses its water upon drying.

Upon more than 20 tests with different $Fe(BF_4)_2$ quantities introduced (from x = 0.25 to x = 1.5), a window was found where a room temperature hysteresis was obtained, for the hydrated version of the compound, $[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x \cdot nH_2O$. From $x_{introduced}$ = 0.775 to 1, properties at room temperature could be obtained, though without precise measurements it remains unclear what precise value returns the largest and/or best centered hysteresis.

However, no good synthesis ratio has been found for the non-hydrated version, $[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x$. This is part of the reason why this compound was, in the end, not considered for the final hybrid coating. Another reason for this is that it is very poorly dispersible in the polyurethane suspension, which makes it even more difficult to obtain homogeneous coatings. Finally, it seems to be particularly sensitive to the different compounds present in the polyurethane.

Still, it is important to mention this compound because the study of its interactions with the polyurethane dispersion and the attempts made to negate these interactions proved very useful for the final hybrid SCO/PU material. These interactions and attempts are explored in a further section.

Trials for piezochromic coatings were conducted using $[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x$ as SCO compound which, as a reminder, presented more desirable properties than $[Fe(NH_2trz)_3]Br_2 \cdot nH_2O$.

Here, it was initially impossible to obtain a coating with any SCO properties since, as the coating dried, it deteriorated into the LS state (displayed on Figure 108).

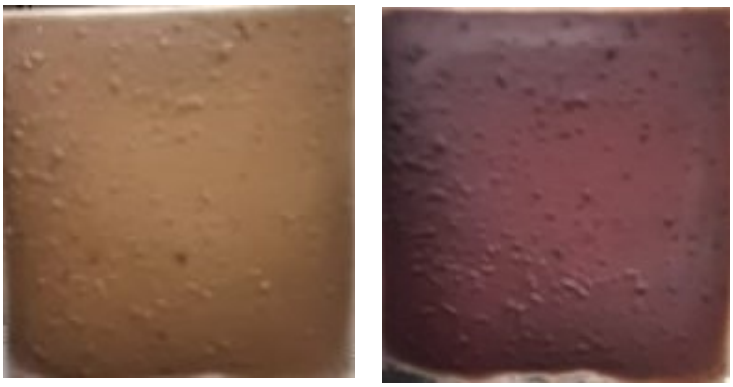
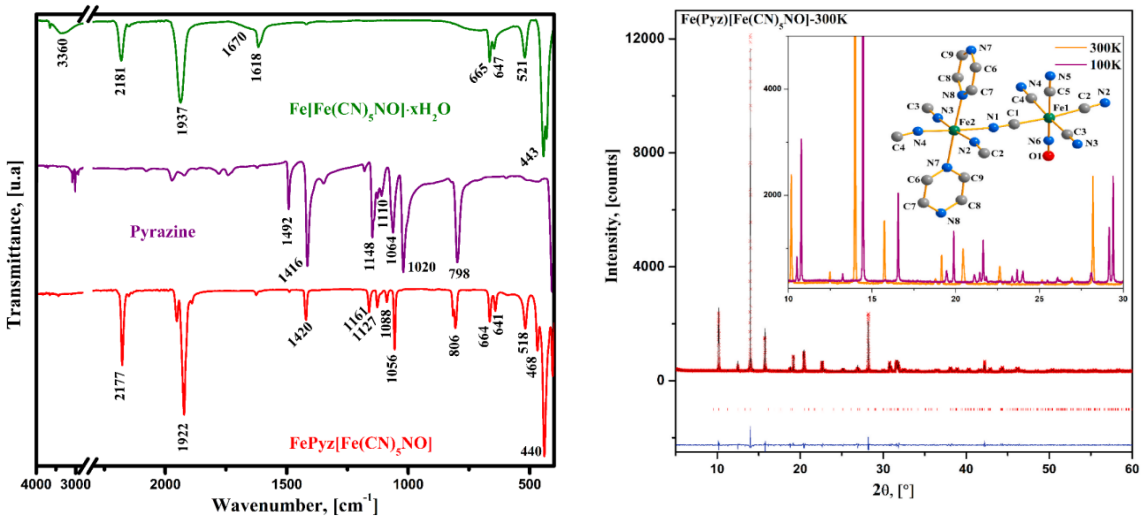


Figure 108 $[Fe(NH_2trz)_3](NO_3)_{2-x}(BF_4)_x$ coating a) in the HS state, wet b) in the final unchangeable LS state after drying.

17.9 Characterization data for Fe(pyrazine)[Fe(CN)$_5$NO]



Right : XRD data from [88]

Left) FTIR data from [88], Fe(pyrazine)[Fe(CN)$_5$NO] displayed in red, the precursor in green.

17.10 DSC measurement of Fe(NH$_2$trz)$_3$]Br$_2$.nH$_2$O-PU hybrid

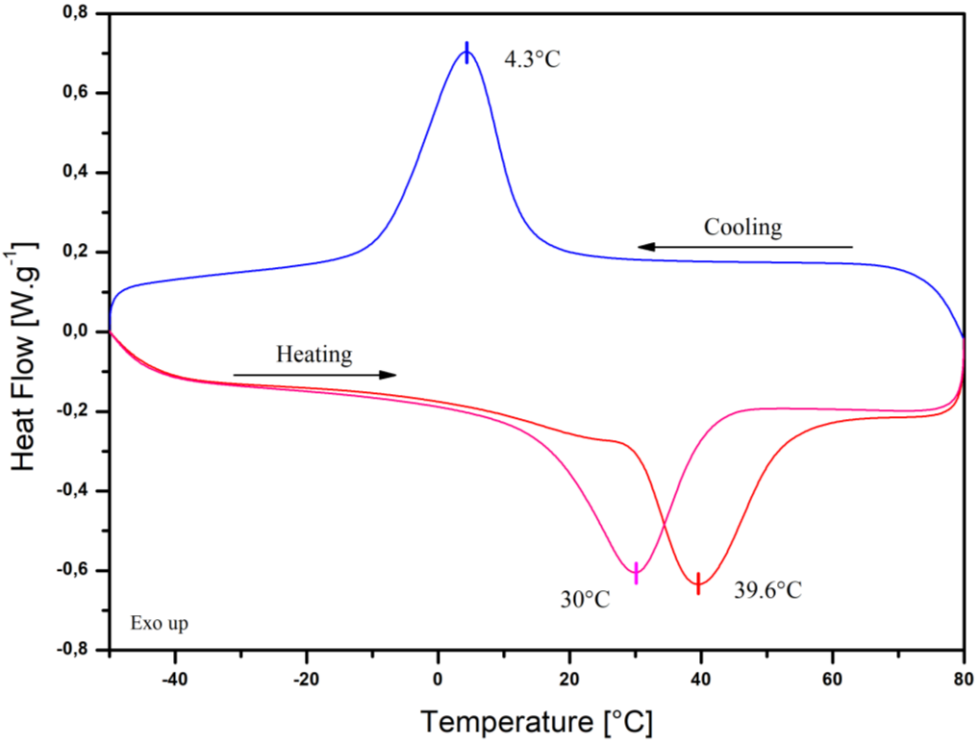
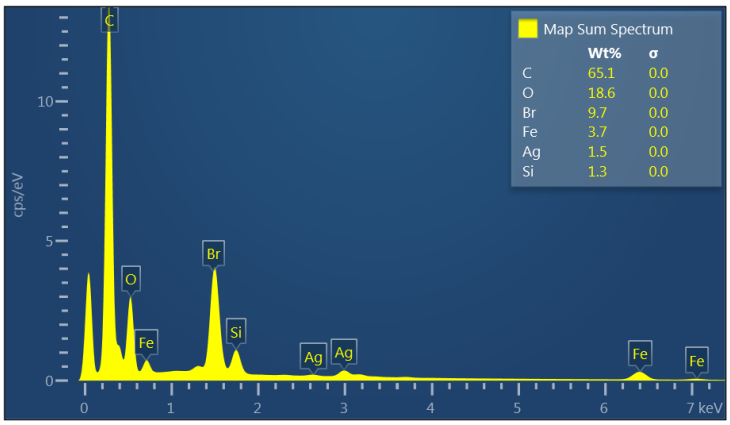
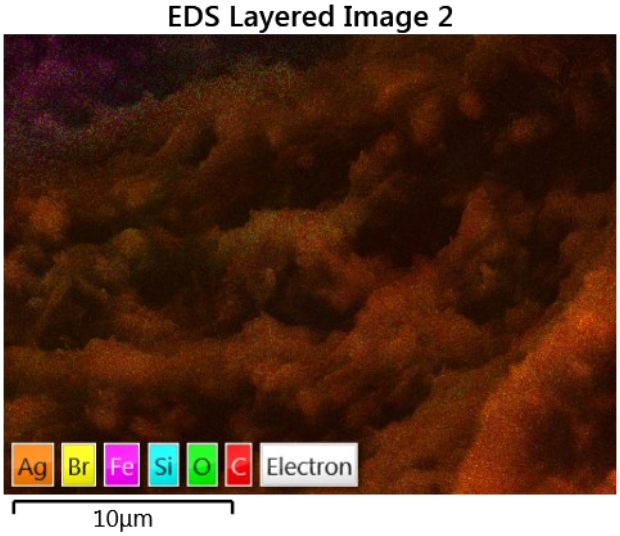


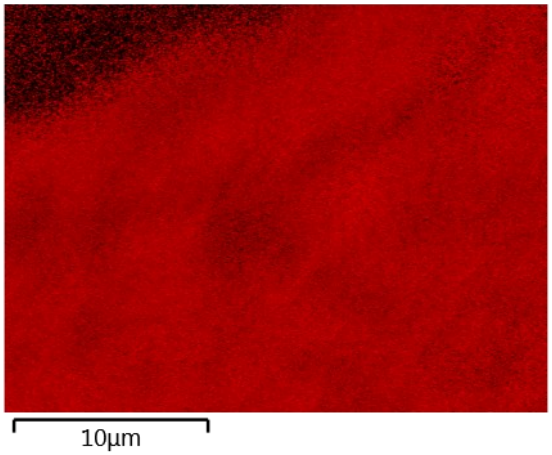
Figure 109 DSC measurement of PU+[Fe(NH$_2$trz)$_3$]Br$_2$·nH$_2$O under N$_2$ atmosphere (100ml/min) at a rate of 10°C/min

17.11 EDX images of Fe(NH$_2$trz)$_3$]Br$_2$.nH$_2$O-PU hybrid

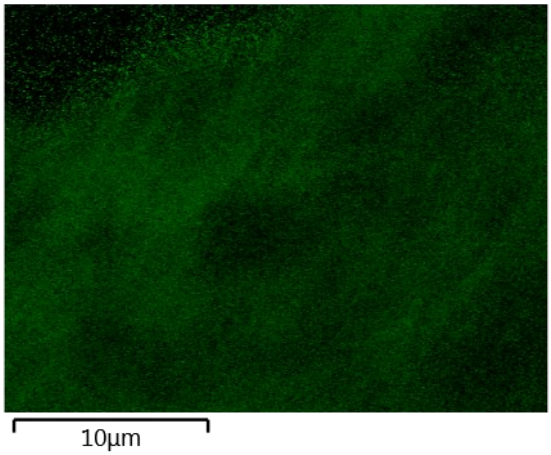




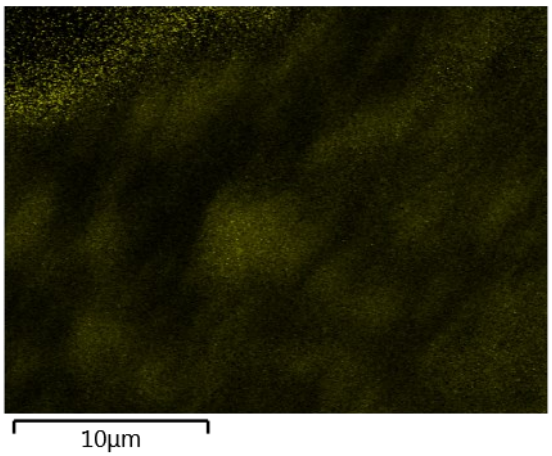
C Wt%



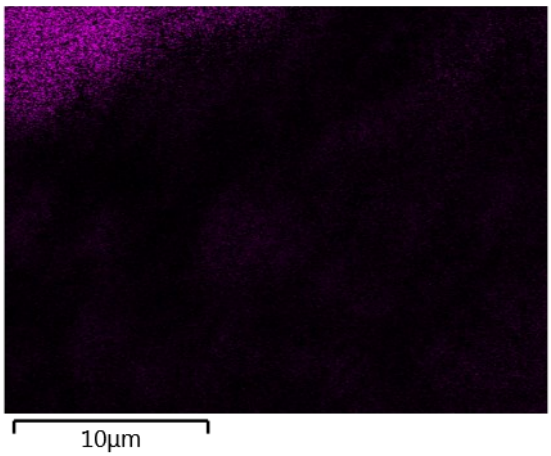
O Wt%



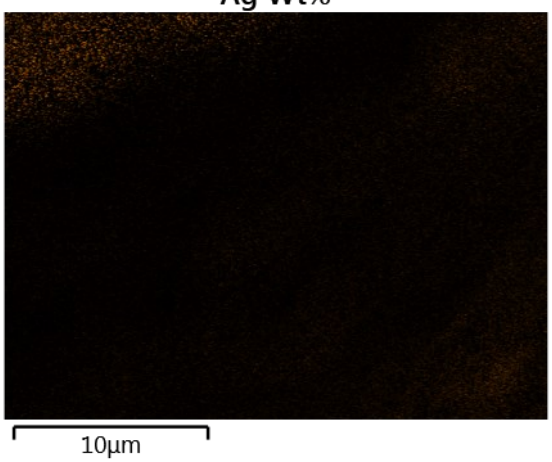
Br Wt%



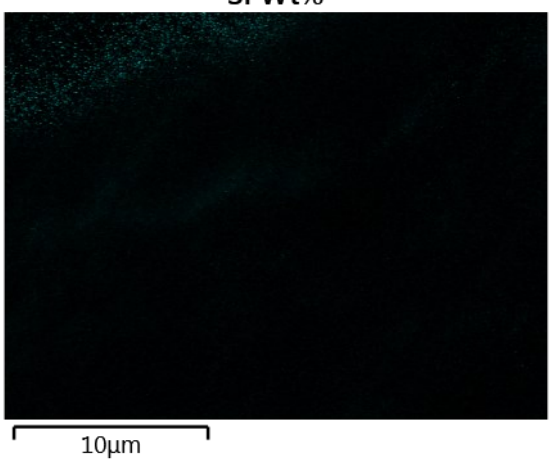
Fe Wt%



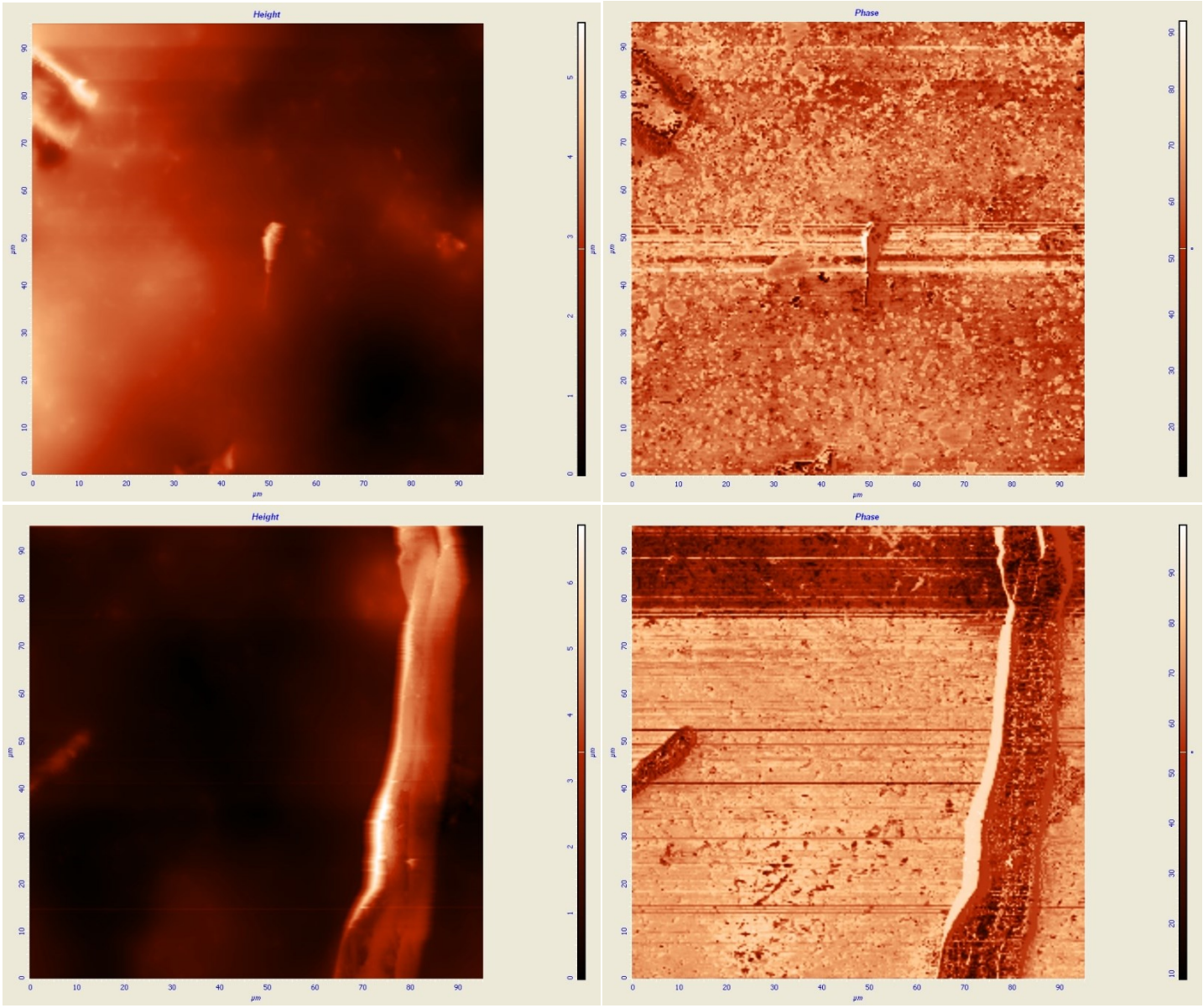
Ag Wt%

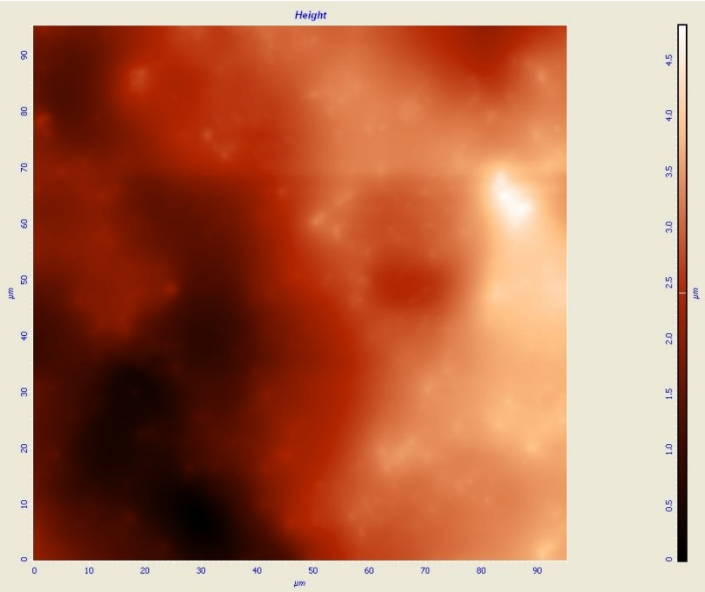


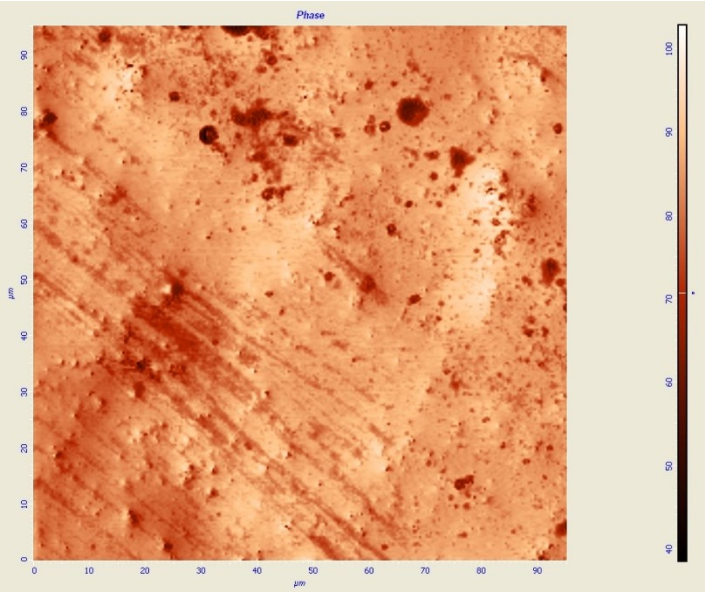
Si Wt%

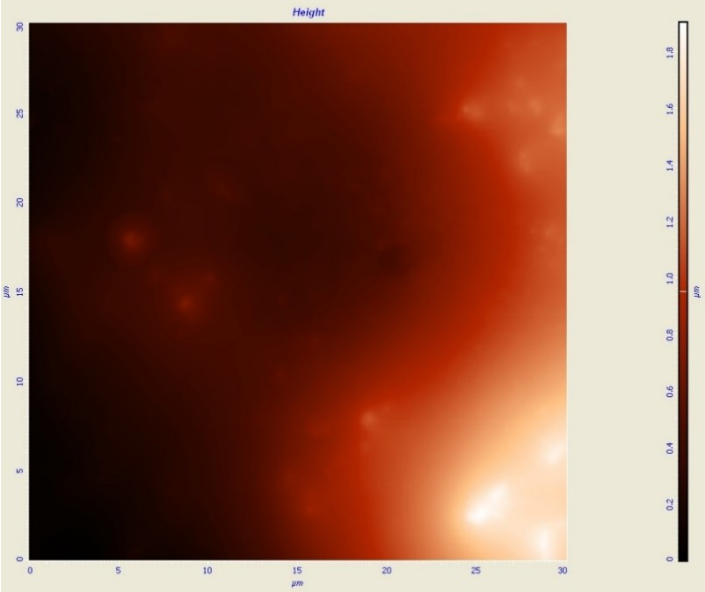


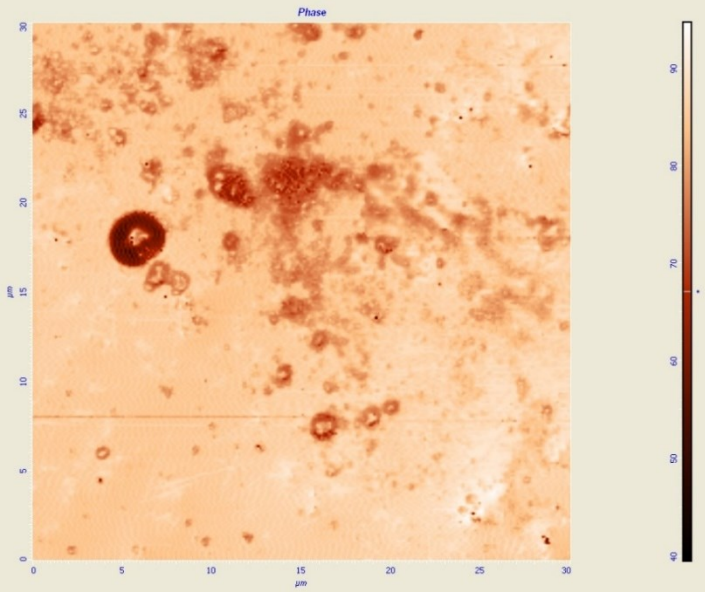
17.12 AFM images of $Fe(NH_2trz)_3]Br_2.nH_2O$-PU-UV hybrid



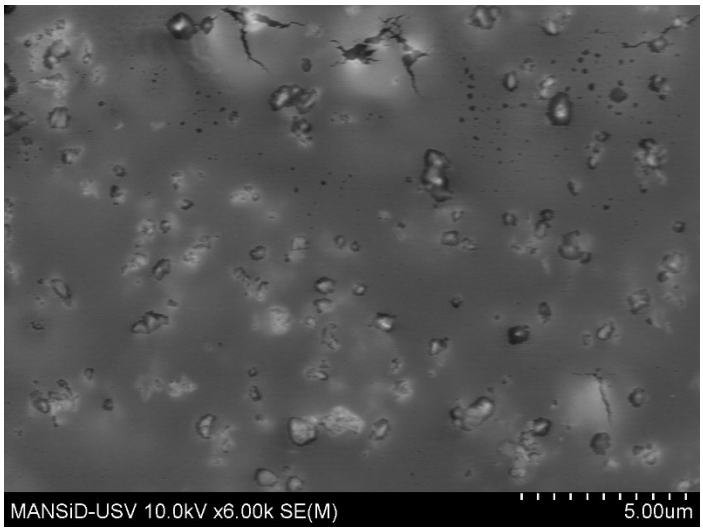

Height
µm
µm

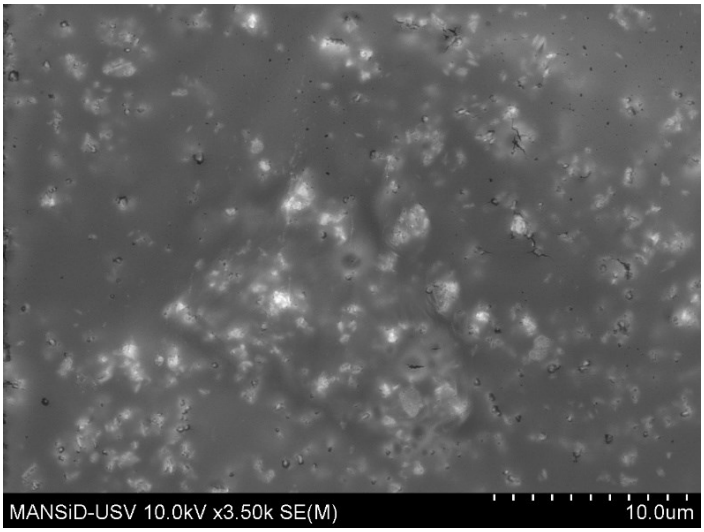

Phase
µm
°

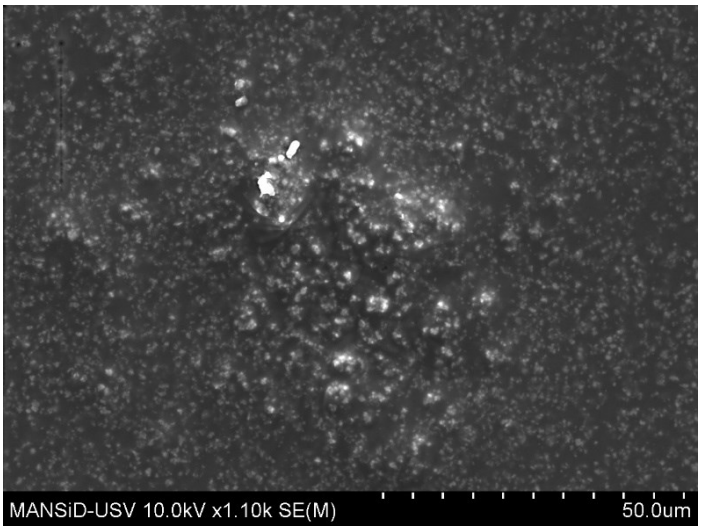

Height
µm
µm

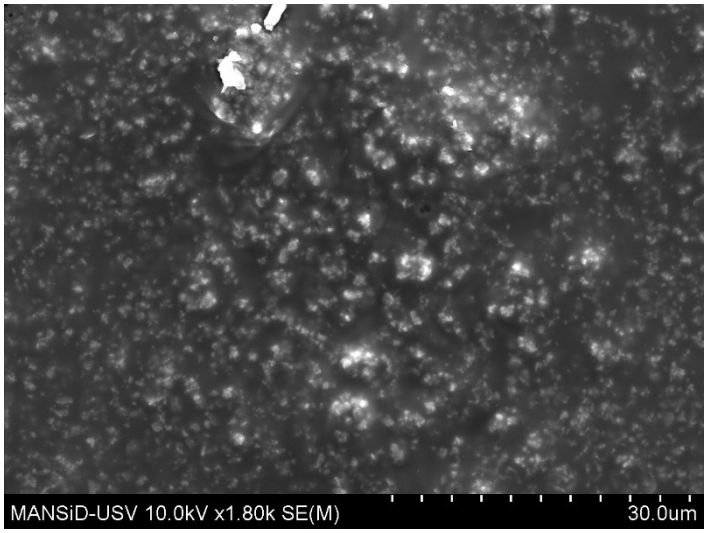

Phase
µm
°

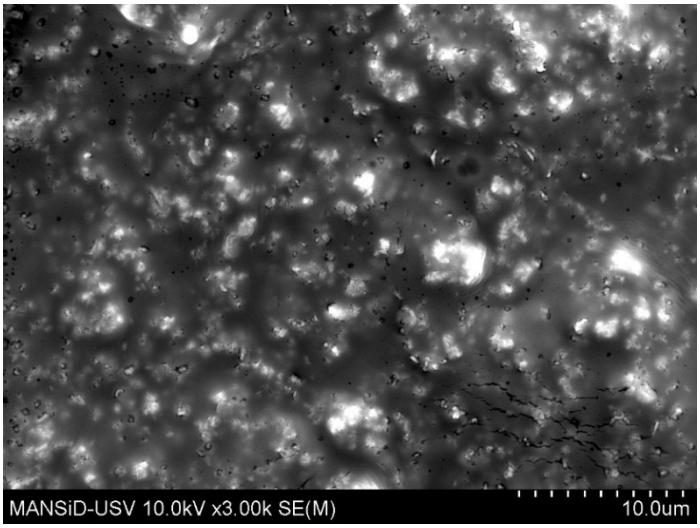
17.13SEM images of $Fe(NH_2trz)_3]Br_2.nH_2O$-PU hybrid (D15)

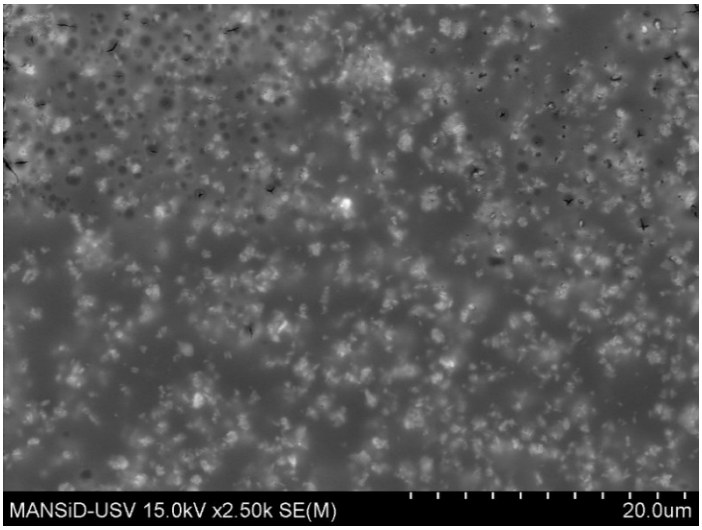


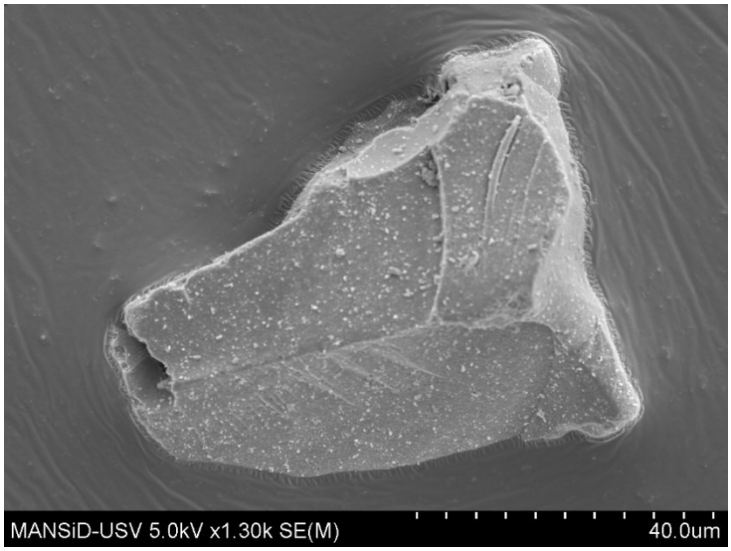


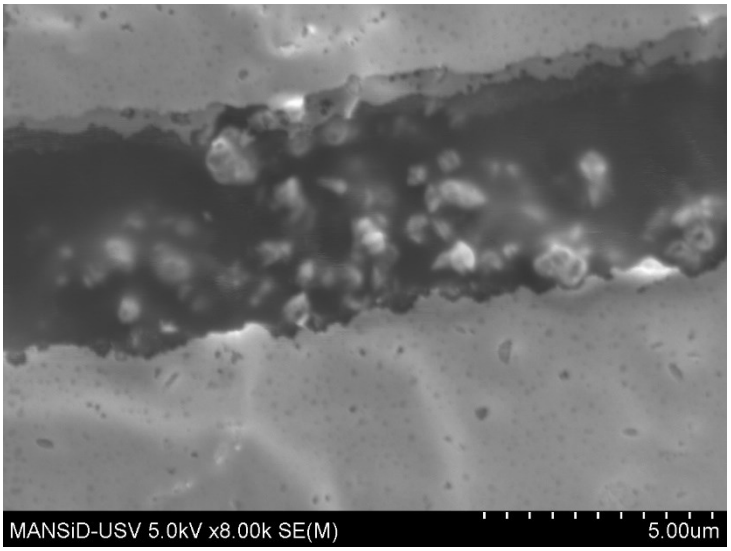


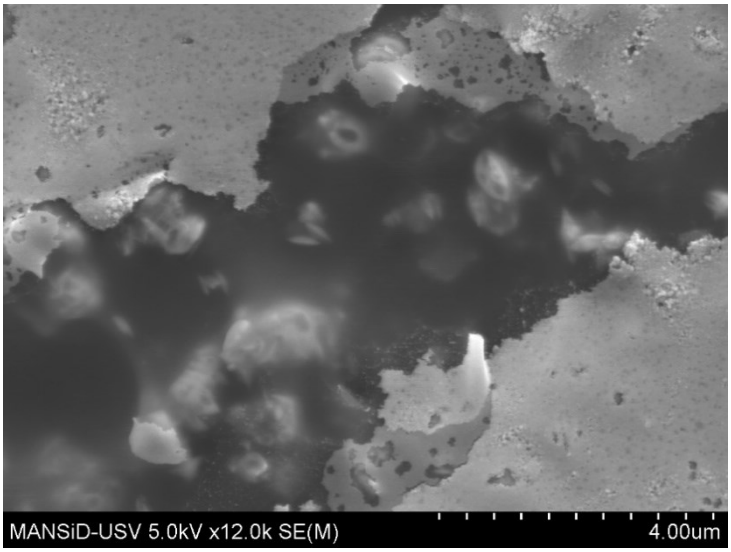


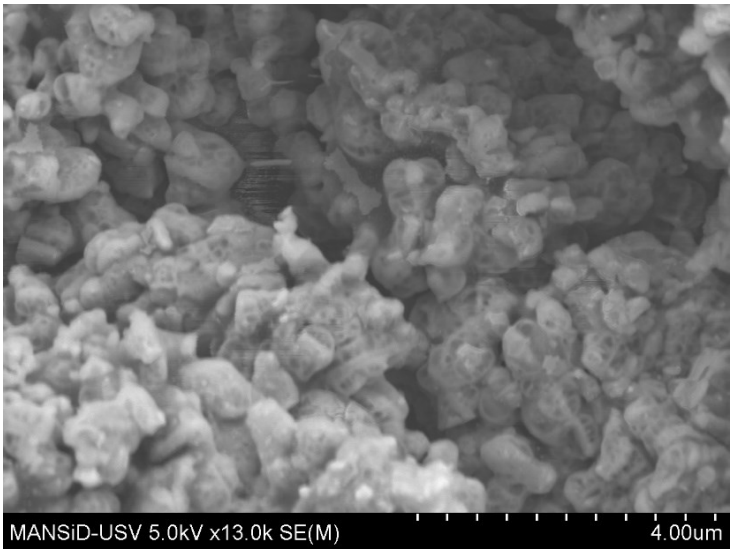


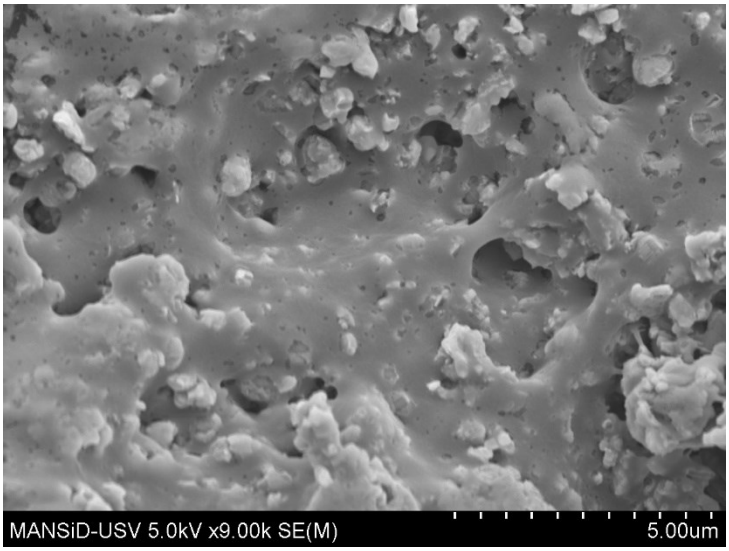


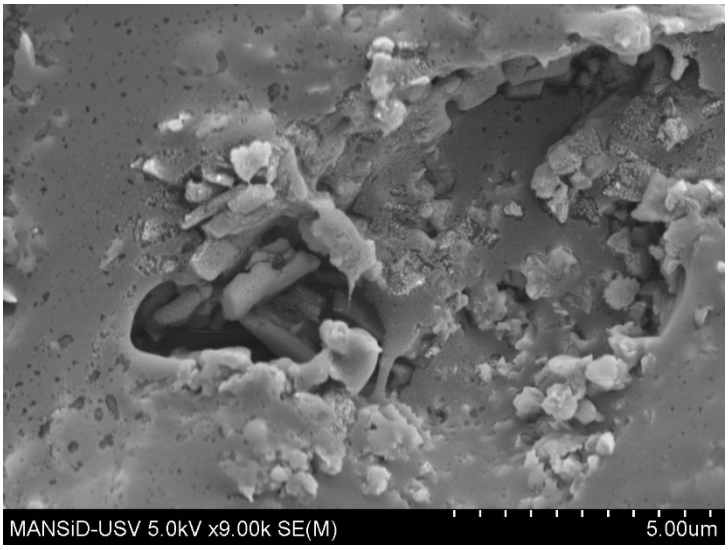

MANSiD-USV 5.0kV x1.30k SE(M)
40.0um

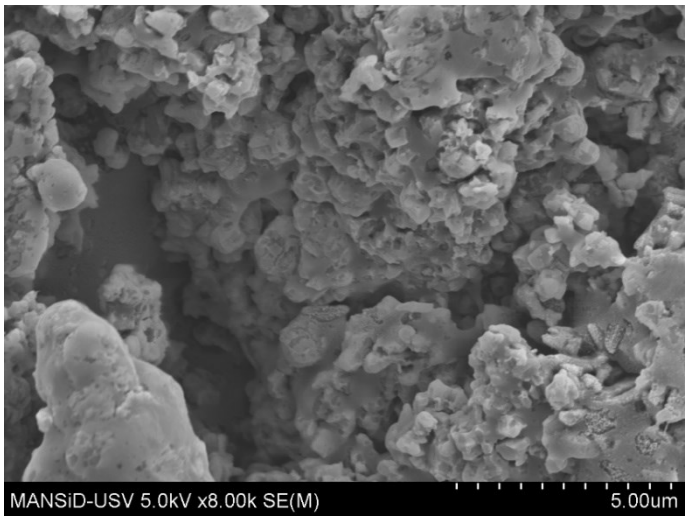

MANSiD-USV 5.0kV x8.00k SE(M)
5.00um

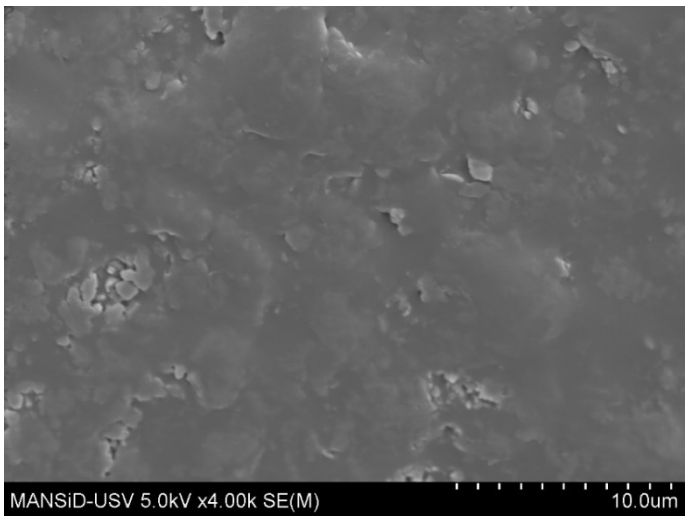

MANSiD-USV 5.0kV x12.0k SE(M)
4.00um

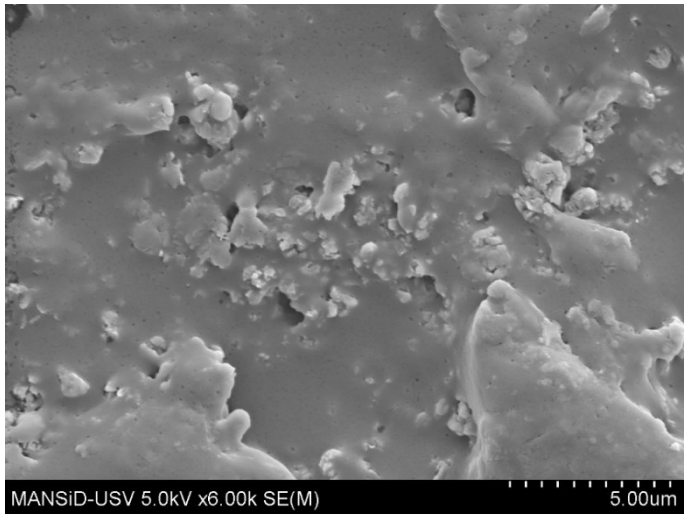

MANSiD-USV 5.0kV x13.0k SE(M)
4.00um

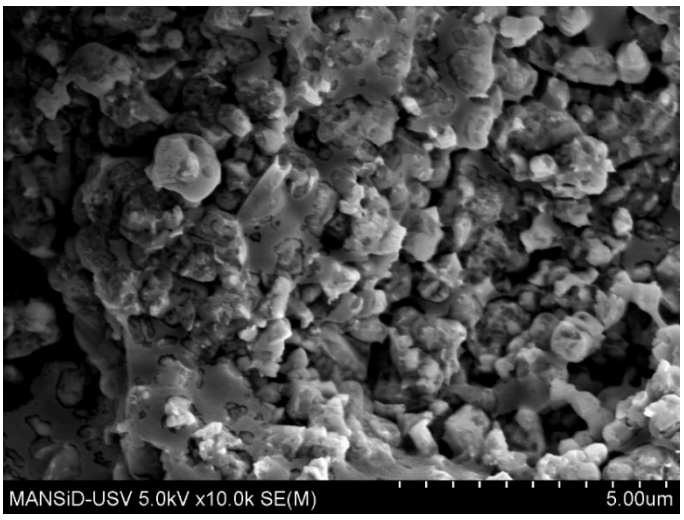

MANSiD-USV 5.0kV x9.00k SE(M)
5.00um

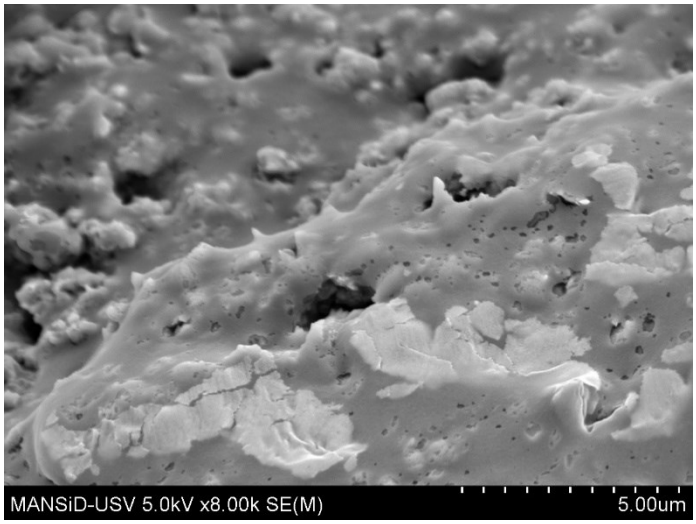

MANSiD-USV 5.0kV x9.00k SE(M)
5.00um

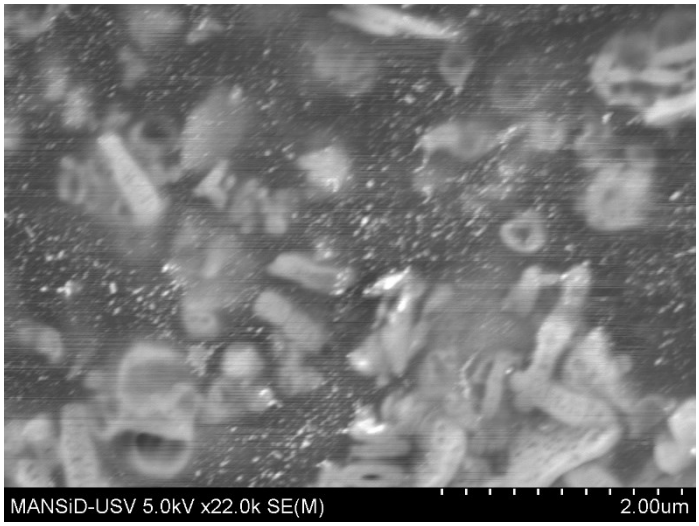

MANSiD-USV 5.0kV x8.00k SE(M)
5.00um

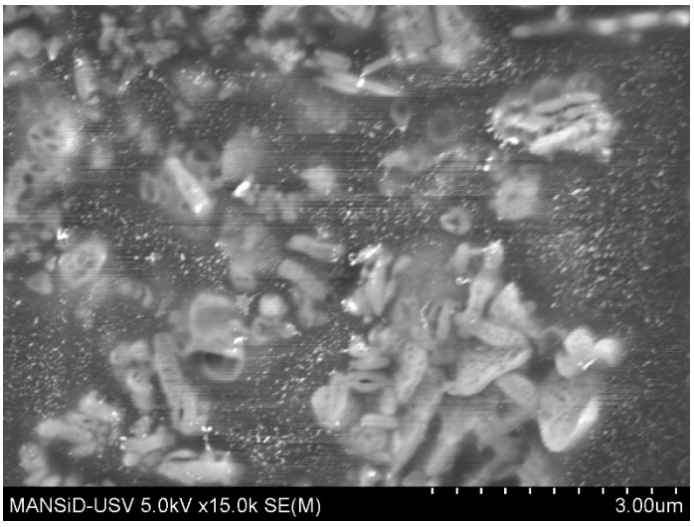

MANSiD-USV 5.0kV x4.00k SE(M)
10.0um

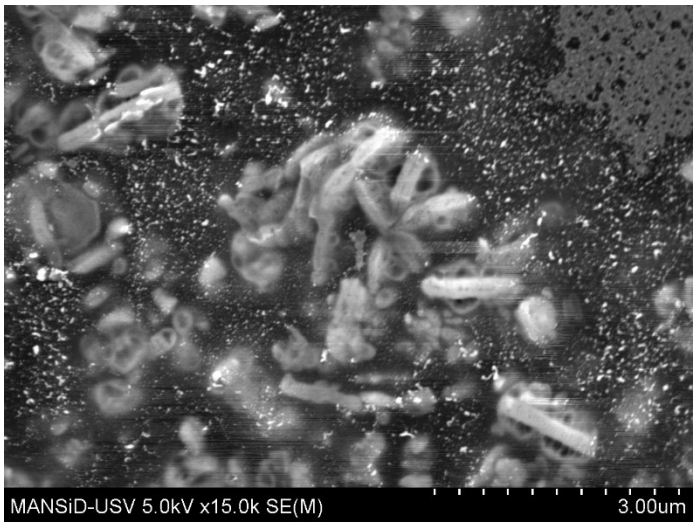

MANSiD-USV 5.0kV x6.00k SE(M)
5.00um

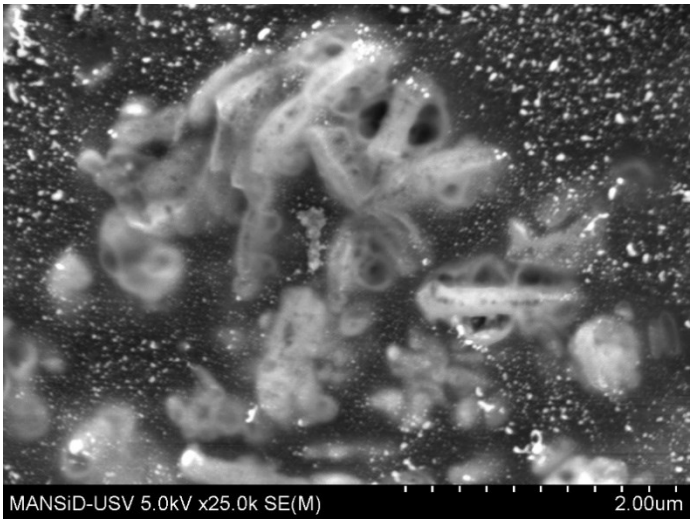

MANSiD-USV 5.0kV x10.0k SE(M)
5.00um

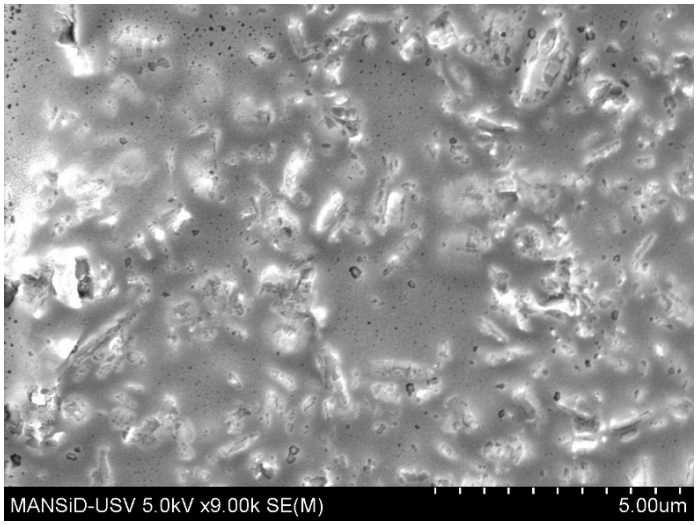

MANSiD-USV 5.0kV x8.00k SE(M)
5.00um

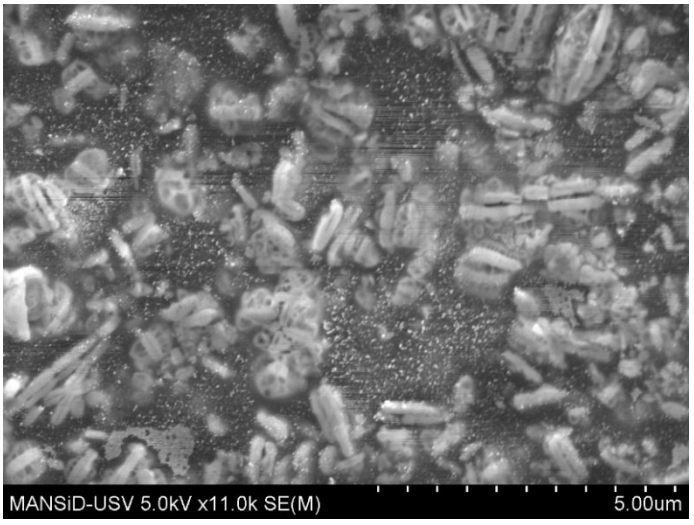

MANSiD-USV 5.0kV x22.0k SE(M)
2.00um

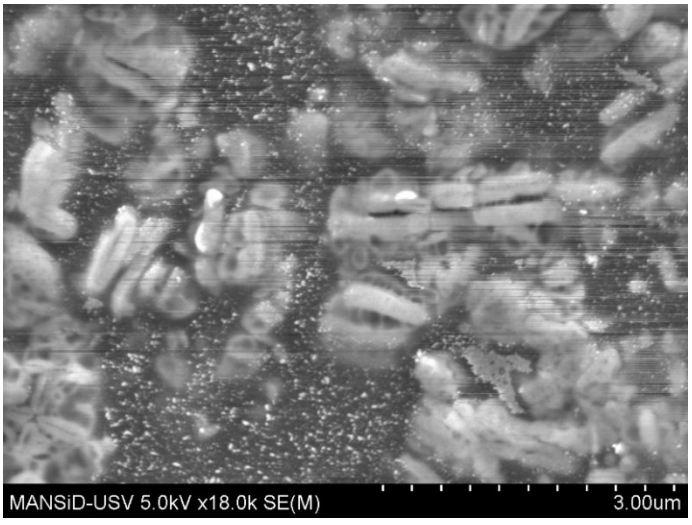

MANSiD-USV 5.0kV x15.0k SE(M)
3.00um

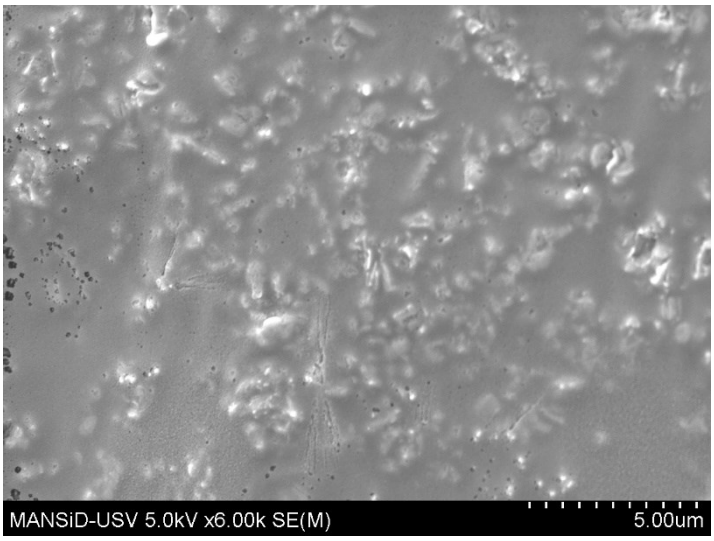

MANSiD-USV 5.0kV x15.0k SE(M)
3.00um

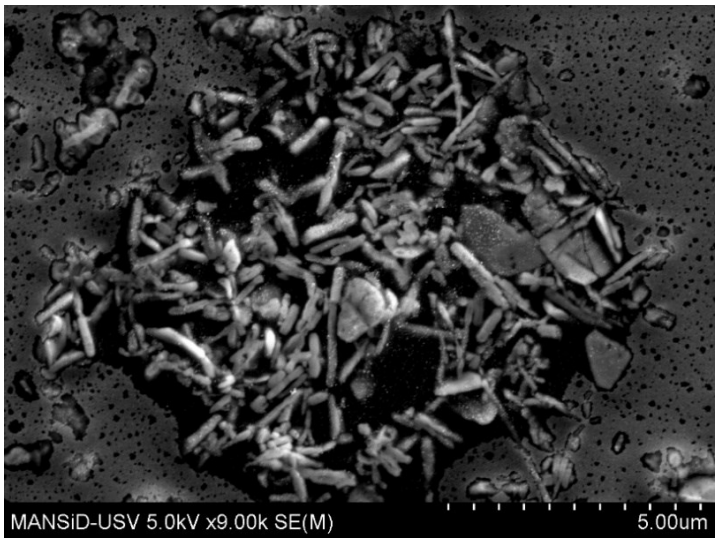

MANSiD-USV 5.0kV x25.0k SE(M)
2.00um

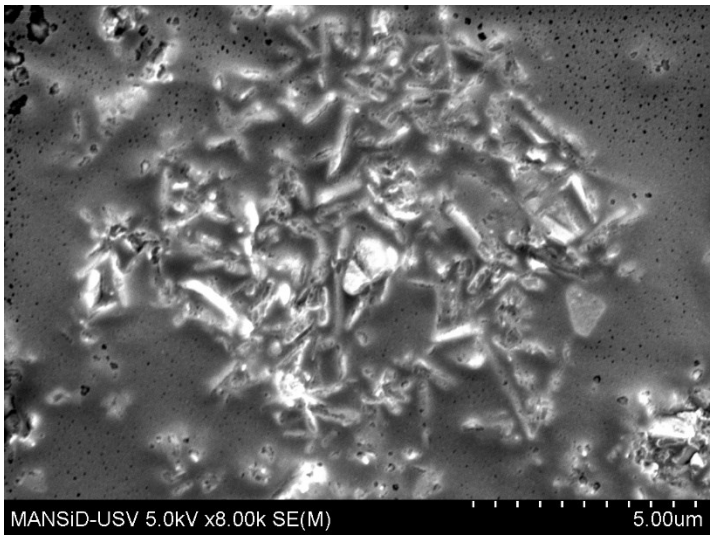

MANSiD-USV 5.0kV x9.00k SE(M)
5.00um


MANSiD-USV 5.0kV x11.0k SE(M)
5.00um

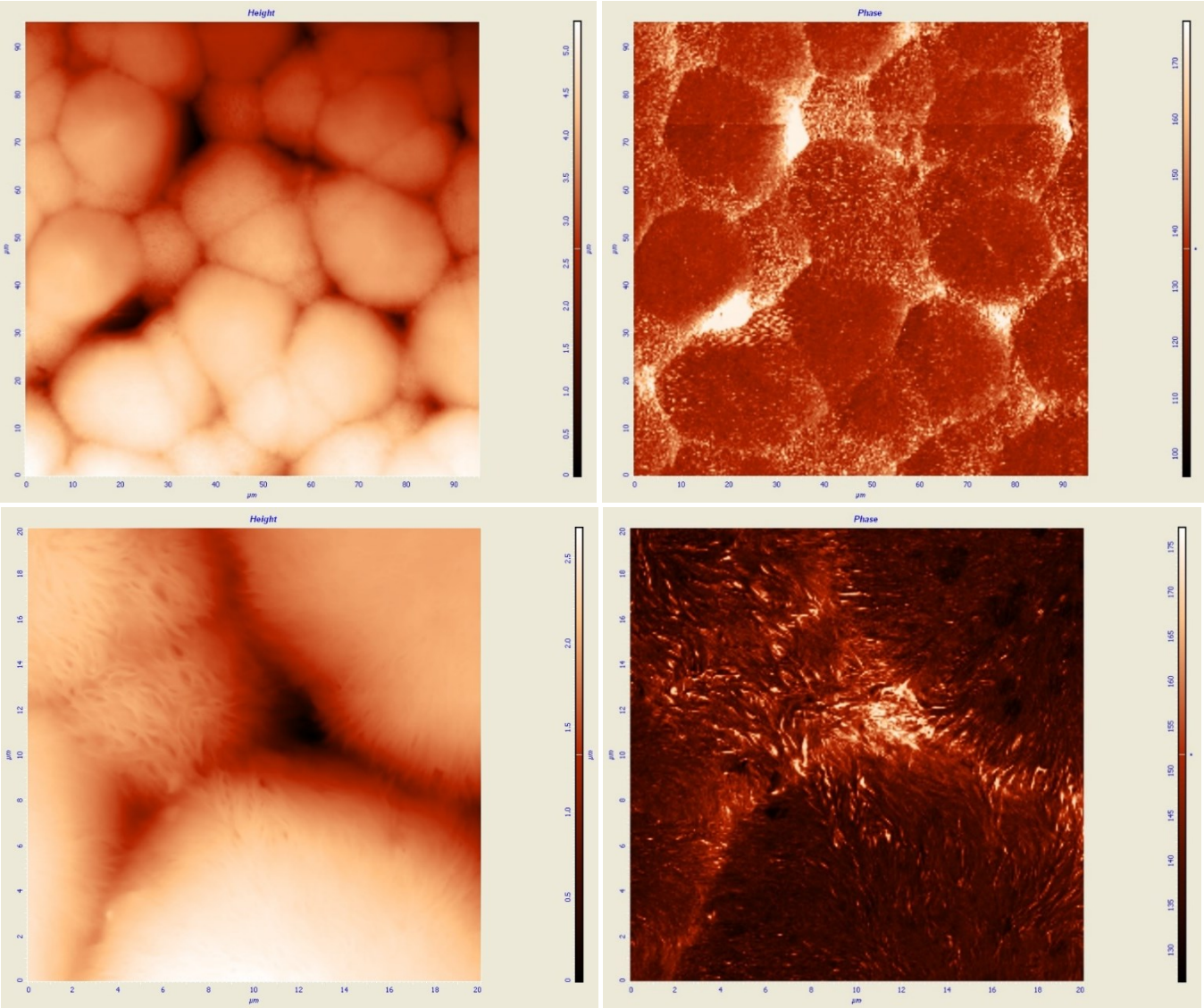

MANSiD-USV 5.0kV x18.0k SE(M)
3.00um

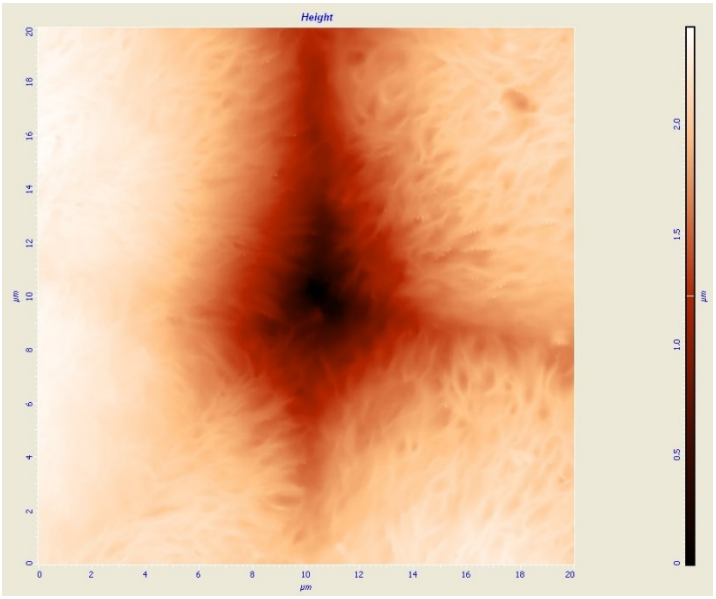

MANSiD-USV 5.0kV x6.00k SE(M)
5.00um

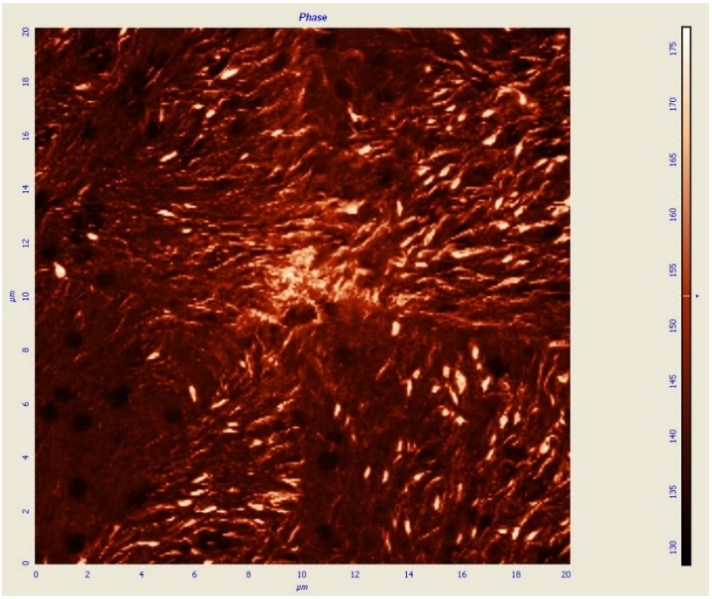

MANSiD-USV 5.0kV x9.00k SE(M)
5.00um

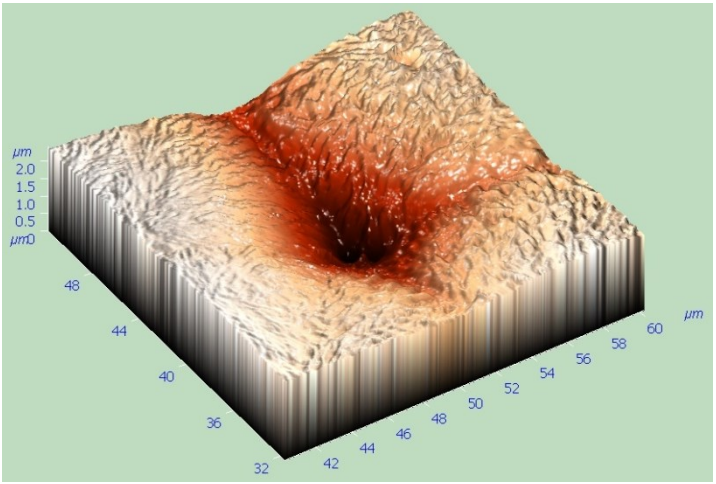

MANSiD-USV 5.0kV x8.00k SE(M)
5.00um

17.14 AFM images of Fe(pyrazine)[$Fe(CN)_5NO$]-PU-UV hybrid









17.15 Python and R code for impact detection

17.15.1 Python Code to read images and display results

```
import numpy as np
import cv2 as cv
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import imageio

#%% Train image for impact detection
################################################
#Reading blacked-out image for training machine learning models (impact detection) + writing info as .csv file
Train = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG3B.jpg')
```

```
x = np.float32(Train)
combined = x[:,:,0] + x[:,:,1] + x[:,:,2]
rows, cols = np.where((np.logical_and(combined>=0, combined<=30)))

z = np.zeros(x.shape[:-1] + (2,), dtype=x.dtype)
imgs_out = np.concatenate((x, z), axis=-1)

imgs_out[rows,cols,-1] = 1
res = imgs_out.reshape((-1,5))

df = pd.DataFrame(res)
df.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/Train.csv', index = False)


#Reading full image for training machine learning models (impact detection) +
writing data as .csv file
Train2 = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/RBG3.jpg')
x2 = np.float32(Train2)
combined2 = x2[:,:,0] + x2[:,:,1] + x2[:,:,2]
rows2, cols2 = np.where((np.logical_and(combined2 >= 0, combined2 <= 30)))

z2 = np.zeros(x2.shape[:-1] + (2,), dtype=x2.dtype)
imgs2 = np.concatenate((x2, z2), axis=-1)

imgs2[rows2,cols2,-1] = 1
res2 = imgs2.reshape((-1,5))

df2 = pd.DataFrame(res2)
df2.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/Train2.csv', index = False)

#%% Test images for impact detection
############################################################
# Image 1
#Reading blacked-out image for testing the machine learning models (impact
detection) + writing data as .csv file
Test11 = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/3B.jpg')

x3 = np.float32(Test11)
combined3 = x3[:,:,0] + x3[:,:,1] + x3[:,:,2]
rows3, cols3 = np.where((np.logical_and(combined3>=0, combined3<=30)))

z3 = np.zeros(x3.shape[:-1] + (2,), dtype=x3.dtype)
imgs3 = np.concatenate((x3, z3), axis=-1)

imgs3[rows3,cols3,-1] = 1
res3 = imgs3.reshape((-1,5))

df3 = pd.DataFrame(res3)
df3.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/Test.csv', index = False)
```

```
#Reading full image for testing the machine learning models (impact detection)
+ writing data as .csv file
Test12     =     cv.imread('C:/Users/InvitXCY2/OneDrive     -     UCL/Bureau/SCO
memoire/Pressure measurements/3.jpg')
x4 = np.float32(Test12)
res4 = x4.reshape((-1,3))

df4 = pd.DataFrame(res4)
df4.to_csv('C:/Users/InvitXCY2/OneDrive   -   UCL/Bureau/SCO   memoire/Pressure
measurements/Test2.csv', index = False)

##############################################################
# Image 2
#Reading blacked-out image for testing the machine learning models (impact
detection) + writing data as .csv file
Test21     =     cv.imread('C:/Users/InvitXCY2/OneDrive     -     UCL/Bureau/SCO
memoire/Pressure measurements/2B.jpg')

x3 = np.float32(Test21)
combined3 = x3[:,:,0] + x3[:,:,1] + x3[:,:,2]
rows3, cols3 = np.where((np.logical_and(combined3>=0, combined3<=30)))

z3 = np.zeros(x3.shape[:-1] + (2,), dtype=x3.dtype)
imgs3 = np.concatenate((x3, z3), axis=-1)

imgs3[rows3,cols3,-1] = 1
res3 = imgs3.reshape((-1,5))

df3 = pd.DataFrame(res3)
df3.to_csv('C:/Users/InvitXCY2/OneDrive   -   UCL/Bureau/SCO   memoire/Pressure
measurements/Test_2.csv', index = False)


#Reading full image for testing the machine learning models (impact detection)
+ writing data as .csv file
Test22     =     cv.imread('C:/Users/InvitXCY2/OneDrive     -     UCL/Bureau/SCO
memoire/Pressure measurements/2.jpg')
x4 = np.float32(Test22)
res4 = x4.reshape((-1,3))

df4 = pd.DataFrame(res4)
df4.to_csv('C:/Users/InvitXCY2/OneDrive   -   UCL/Bureau/SCO   memoire/Pressure
measurements/Test2_2.csv', index = False)

# Image 3
#Reading blacked-out image for testing the machine learning models (impact
detection) + writing data as .csv file
Test31     =     cv.imread('C:/Users/InvitXCY2/OneDrive     -     UCL/Bureau/SCO
memoire/Pressure measurements/1B.jpg')

x3 = np.float32(Test31)
combined3 = x3[:,:,0] + x3[:,:,1] + x3[:,:,2]
rows3, cols3 = np.where((np.logical_and(combined3>=0, combined3<=30)))

z3 = np.zeros(x3.shape[:-1] + (2,), dtype=x3.dtype)
imgs3 = np.concatenate((x3, z3), axis=-1)
```

```
imgs3[rows3,cols3,-1] = 1
res3 = imgs3.reshape((-1,5))

df3 = pd.DataFrame(res3)
df3.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test_3.csv', index = False)


#Reading full image for testing the machine learning models (impact detection) + writing data as .csv file
Test32 = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/1.jpg')
x4 = np.float32(Test32)
res4 = x4.reshape((-1,3))

df4 = pd.DataFrame(res4)
df4.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test2_3.csv', index = False)


#%%
# Displaying of the Ensemble model Results for regular RGB data for the 3 tested images
one = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/1ens.csv')
x5 = np.float32(one)
x5 = x5.reshape(237,288)

for i in range(237):
    for j in range(288):
        if x5[i,j] == 1:
            Test32[i,j,0] = 0
            Test32[i,j,1] = 255
            Test32[i,j,2] = 0

plt.imshow(Test32)


two = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/2ens.csv')
x5 = np.float32(two)
x5 = x5.reshape(191,294)

for i in range(191):
    for j in range(294):
        if x5[i,j] == 1:
            Test22[i,j,0] = 0
            Test22[i,j,1] = 255
            Test22[i,j,2] = 0

plt.imshow(Test22)


three = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/3ens.csv')
x5 = np.float32(three)
x5 = x5.reshape(199,206)
```

```
for i in range(199):
    for j in range(206):
        if x5[i,j] == 1:
            Test12[i,j,0] = 0
            Test12[i,j,1] = 255
            Test12[i,j,2] = 0

plt.imshow(Test12)


#%% Prob of impact (Random Forest model)
# Probability of impact map Image 1
i1 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/rf/1rf_prob.csv')
i1 = np.float32(i1)

i1 = i1.reshape(237,288)
plt.imshow(i1)
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

# Probability of impact map Image 2
i2 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/rf/2rf_prob.csv')
i2 = np.float32(i2)

i2 = i2.reshape(191,294)
plt.imshow(i2)
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

# Probability of impact map Image 3
i3 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/rf/3rf_prob.csv')
i3 = np.float32(i3)

i3 = i3.reshape(199,206)
plt.imshow(i3)
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

#%% Prob of impact (k-NN model)
# Probability of impact map Image 1
i1 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/knn/1knn_Prob.csv')
i1 = np.float32(i1)

i1 = i1.reshape(237,288)
plt.imshow(i1, cmap = "copper")
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

# Probability of impact map Image 2
i2 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/knn/2knn_Prob.csv')
i2 = np.float32(i2)
```

```
i2 = i2.reshape(191,294)
plt.imshow(i2, cmap = "copper")
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

# Probability of impact map Image 3
i3 = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/knn/3knn_prob.csv')
i3 = np.float32(i3)

i3 = i3.reshape(199,206)
plt.imshow(i3, cmap = "copper")
plt.colorbar(ticks= (50,100,150,200,250))
plt.axis('off')

# Displaying of the Ensemble model Results for normalized RGB data for the 3 tested images
one = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/N-1ens.csv')
x5 = np.float32(one)
x5 = x5.reshape(237,288)

for i in range(237):
    for j in range(288):
        if x5[i,j] == 1:
            Test32[i,j,0] = 0
            Test32[i,j,1] = 255
            Test32[i,j,2] = 0

plt.imshow(cv.cvtColor(Test32, cv.COLOR_BGR2RGB))
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/one.png", dpi=300)

two = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/N-2ens.csv')
x5 = np.float32(two)
x5 = x5.reshape(191,294)

for i in range(191):
    for j in range(294):
        if x5[i,j] == 1:
            Test22[i,j,0] = 0
            Test22[i,j,1] = 255
            Test22[i,j,2] = 0

plt.imshow(cv.cvtColor(Test22, cv.COLOR_BGR2RGB))
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/two.png", dpi=300)

three = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/N-3ens.csv')
x5 = np.float32(three)
x5 = x5.reshape(199,206)
```

```
for i in range(199):
    for j in range(206):
        if x5[i,j] == 1:
            Test12[i,j,0] = 0
            Test12[i,j,1] = 255
            Test12[i,j,2] = 0

plt.imshow(cv.cvtColor(Test12, cv.COLOR_BGR2RGB))
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/three.png", dpi=300)

#%% Pressure mapping over the entire coating image
tot = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG3.jpg')
tot1 = np.float32(tot)
tot1 = tot1.reshape((-1,3))
tot2 = pd.DataFrame(tot1)

tot2.to_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/PMAP.csv')

four = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/YMAP.csv')
four = np.float32(four)

four = four.reshape(250,332)
plt.imshow(four, cmap = "plasma")
plt.colorbar(ticks= (5,122,250))
plt.axis('off')

plt.imshow(four, cmap = "gist_heat")
plt.colorbar(ticks= (0,50,100,200,250))
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/combi.png", dpi=300)

#%% Combine Impact detection with Pressure determination
# Displaying Impact detection results on the coating
tain = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/Tens.csv')
x5 = np.float32(tain)
x5 = x5.reshape(250,332)
tain1 = cv.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG3.jpg')
for i in range(250):
    for j in range(332):
        if x5[i,j] == 1:
            tain1[i,j,0] = 0
            tain1[i,j,1] = 255
            tain1[i,j,2] = 0
        else:
            tain1[i,j,0] = tain1[i,j,0]
            tain1[i,j,1] = tain1[i,j,1]
            tain1[i,j,2] = tain1[i,j,2]

plt.imshow(cv.cvtColor(tain1, cv.COLOR_BGR2RGB))
```

```
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/impactDet.png", dpi=300)


#### Displaying the pressures predicted only for the pixels which were considered as impacted by the ensemble model trained
four = pd.read_csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/YMAP.csv')
four = np.float32(four)
four = four.reshape(250,332)

cmap = plt.cm.gist_heat
norm = plt.Normalize(four.min(), four.max())
rgba = cmap(norm(four))
ttt = norm(tain1)
for i in range(250):
    for j in range(332):
        if x5[i,j] != 1:
            rgba[i,j,0] = ttt[i,j,2]
            rgba[i,j,1] = ttt[i,j,1]
            rgba[i,j,2] = ttt[i,j,0]
fig, ax = plt.subplots()
ax.imshow(rgba)
im = ax.imshow(four, visible=False, cmap=cmap)
fig.colorbar(im)
plt.axis('off')
plt.savefig("C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/combi.png", dpi=300)
plt.show()
```

17.15.2 R code for Machine learning model training

```
library(caret)
library(tidyverse)

#Reading and creation of the training data dataframe
df_black <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Train.csv')
df2_all <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Train2.csv')
Train <- data.frame(one = df2_all$X0, two = df2_all$X1, three = df2_all$X2, Binom = df_black$X4)

#Logistic regression model trained for impact detection and tested on 3 images
control <- trainControl(method = "cv", number = 10, p = .9)
train_glm <- train(as.factor(Binom) ~ one + two + three,
                   method = "glm",
                   trControl = control,
                   data = Train)

#Test for 3,3B
Test_black3 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test.csv')
Test_all3 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test2.csv')
```

```
Test3 <- data.frame(one = Test_all3$X0, two = Test_all3$X1, three =
Test_all3$X2, Binom = Test_black3$X4)
Y_hat3glm <- predict(train_glm, Test3[,1:3])
Y_hat3glm
cm3glm <- confusionMatrix(Y_hat3glm, factor(Test3$Binom))
cm3glm$overall["Accuracy"]

g3 <- cm3glm$overall["Accuracy"]

write.csv(Y_hat3glm,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/3glm.csv', row.names = F)

#Test for 2, 2B
Test_black2 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/Test_2.csv')
Test_all2 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/Test2_2.csv')
Test2 <- data.frame(one = Test_all2$X0, two = Test_all2$X1, three =
Test_all2$X2, Binom = Test_black2$X4)
Y_hat2glm <- predict(train_glm, Test2[,1:3])
Y_hat2glm
cm2glm <- confusionMatrix(Y_hat2glm, factor(Test2$Binom))
cm2glm$overall["Accuracy"]

g2 <- cm2glm$overall["Accuracy"]

write.csv(Y_hat2glm,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/2glm.csv', row.names = F)

#Test for 1, 1.B
Test_black1 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/Test_3.csv')
Test_all1 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/Test2_3.csv')
Test1 <- data.frame(one = Test_all1$X0, two = Test_all1$X1, three =
Test_all1$X2, Binom = Test_black1$X4)
Y_hat1glm <- predict(train_glm, Test1[,1:3])
Y_hat1glm
cm1glm <- confusionMatrix(Y_hat1glm, factor(Test1$Binom))
cm1glm$overall["Accuracy"]

g1 <- cm1glm$overall["Accuracy"]

write.csv(Y_hat1glm,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/1glm.csv', row.names = F)

#Random Forest model trained for impact detection and tested on 3 images
control <- trainControl(method = "cv", number = 10, p = .9)
train_rf <- train(as.factor(Binom) ~ one + two + three,
                  method = "rf",
                  trControl = control,
                  data = Train)

#Test for 3,3B
Test_black3 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO
memoire/Pressure measurements/Test.csv')
```

```
Test_all3    <-    read.csv('C:/Users/InvitXCY2/OneDrive    -    UCL/Bureau/SCO
memoire/Pressure measurements/Test2.csv')
Test3   <-   data.frame(one   =   Test_all3$X0,   two   =   Test_all3$X1,   three   =
Test_all3$X2, Binom = Test_black3$X4)
Y_hat3rf <- predict(train_rf, Test3[,1:3])
Y_hat3rf
cm3rf <- confusionMatrix(Y_hat3rf, factor(Test3$Binom))
cm3rf$overall["Accuracy"]

r3 <- cm3rf$overall["Accuracy"]

write.csv(Y_hat3rf,'C:/Users/InvitXCY2/OneDrive        -        UCL/Bureau/SCO
memoire/Pressure measurements/3rf.csv', row.names = F)

#Test for 2, 2B
Test_black2    <-    read.csv('C:/Users/InvitXCY2/OneDrive    -    UCL/Bureau/SCO
memoire/Pressure measurements/Test_2.csv')
Test_all2    <-    read.csv('C:/Users/InvitXCY2/OneDrive    -    UCL/Bureau/SCO
memoire/Pressure measurements/Test2_2.csv')
Test2   <-   data.frame(one   =   Test_all2$X0,   two   =   Test_all2$X1,   three   =
Test_all2$X2, Binom = Test_black2$X4)
Y_hat2rf <- predict(train_rf, Test2[,1:3])
Y_hat2rf
cm2rf <- confusionMatrix(Y_hat2rf, factor(Test2$Binom))
cm2rf$overall["Accuracy"]

r2 <- cm2rf$overall["Accuracy"]

write.csv(Y_hat2rf,'C:/Users/InvitXCY2/OneDrive        -        UCL/Bureau/SCO
memoire/Pressure measurements/2rf.csv', row.names = F)

#Test for 1, 1.B
Test_black1    <-    read.csv('C:/Users/InvitXCY2/OneDrive    -    UCL/Bureau/SCO
memoire/Pressure measurements/Test_3.csv')
Test_all1    <-    read.csv('C:/Users/InvitXCY2/OneDrive    -    UCL/Bureau/SCO
memoire/Pressure measurements/Test2_3.csv')
Test1   <-   data.frame(one   =   Test_all1$X0,   two   =   Test_all1$X1,   three   =
Test_all1$X2, Binom = Test_black1$X4)
Y_hat1rf <- predict(train_rf, Test1[,1:3])
Y_hat1rf
cm1rf <- confusionMatrix(Y_hat1rf, factor(Test1$Binom))
cm1rf$overall["Accuracy"]

r1 <- cm1rf$overall["Accuracy"]

write.csv(Y_hat1rf,'C:/Users/InvitXCY2/OneDrive        -        UCL/Bureau/SCO
memoire/Pressure measurements/1rf.csv', row.names = F)

#k-NN model trained for impact detection and tested on 3 images
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(as.factor(Binom) ~ one + two + three,
                   method = "knn",
                   tuneGrid = data.frame(k = c(3,5,7)),
                   trControl = control,
                   data = Train)

#Test for 3,3B
```

```
Test_black3 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test.csv')
Test_all3 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test2.csv')
Test3 <- data.frame(one = Test_all3$X0, two = Test_all3$X1, three = Test_all3$X2, Binom = Test_black3$X4)
Y_hat3knn <- predict(train_knn, Test3[,1:3])
Y_hat3knn
cm3knn <- confusionMatrix(Y_hat3knn, factor(Test3$Binom))
cm3knn$overall["Accuracy"]
k3 <- cm3knn$overall["Accuracy"]

write.csv(Y_hat3knn,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/3knn.csv', row.names = F)

#Test for 2, 2B
Test_black2 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test_2.csv')
Test_all2 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test2_2.csv')
Test2 <- data.frame(one = Test_all2$X0, two = Test_all2$X1, three = Test_all2$X2, Binom = Test_black2$X4)
Y_hat2knn <- predict(train_knn, Test2[,1:3])
Y_hat2knn
cm2knn <- confusionMatrix(Y_hat2knn, factor(Test2$Binom))
cm2knn$overall["Accuracy"]

k2 <- cm2knn$overall["Accuracy"]

write.csv(Y_hat2knn,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/2knn.csv', row.names = F)

#Test for 1, 1.B
Test_black1 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test_3.csv')
Test_all1 <- read.csv('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Test2_3.csv')
Test1 <- data.frame(one = Test_all1$X0, two = Test_all1$X1, three = Test_all1$X2, Binom = Test_black1$X4)
Y_hat1knn <- predict(train_knn, Test1[,1:3])
Y_hat1knn
cm1knn <- confusionMatrix(Y_hat1knn, factor(Test1$Binom))
cm1knn$overall["Accuracy"]

k1 <- cm1knn$overall["Accuracy"]

write.csv(Y_hat1knn,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/1knn.csv', row.names = F)

#Ensemble model creation and testing on 3 images
#Test for 1,1.B
meanY1 <- (as.numeric(Y_hat1glm)-1 + as.numeric(Y_hat1knn)-1 + as.numeric(Y_hat1rf)-1)/3

meanY1[which(meanY1 >= 1)] <- 1
meanY1[which(meanY1 < 1)] <- 0
```

```
cm1ens <- confusionMatrix(factor(meanY1), factor(Test1$Binom))
cm1ens$overall["Accuracy"]

e1 <- cm1ens$overall["Accuracy"]
write.csv(meanY1,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/ens/1ens.csv', row.names = F)

#Test for 2, 2.B
meanY2    <-    (as.numeric(Y_hat2glm)-1    +    as.numeric(Y_hat2knn)-1    +
as.numeric(Y_hat2rf)-1)/3

meanY2[which(meanY2 >= 1)] <- 1
meanY2[which(meanY2 < 1)] <- 0

cm2ens <- confusionMatrix(factor(meanY2), factor(Test2$Binom))
cm2ens$overall["Accuracy"]

e2 <- cm2ens$overall["Accuracy"]
write.csv(meanY2,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/ens/2ens.csv', row.names = F)

#Test for 3, 3.B
meanY3    <-    (as.numeric(Y_hat3glm)-1    +    as.numeric(Y_hat3knn)-1    +
as.numeric(Y_hat3rf)-1)/3

meanY3[which(meanY3 >= 1)] <- 1
meanY3[which(meanY3 < 1)] <- 0

cm3ens <- confusionMatrix(factor(meanY3), factor(Test3$Binom))
cm3ens$overall["Accuracy"]

e3 <- cm3ens$overall["Accuracy"]
write.csv(meanY3,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure
measurements/ens/3ens.csv', row.names = F)

######
######
####
####
# The exact same methodology was used to train the impact detection models on
the normalized RGB values #
####
####
######
######

#Prediction on the original image (train image) to display combination of impact
detection and pressure prediction
Train1 = data.frame(one = df2_all$X0, two = df2_all$X1, three = df2_all$X2)
# Logistic regression model
Y_hatTglm <- predict(train_glm, Train1)
Y_hatTglm
cmTglm <- confusionMatrix(Y_hatTglm, factor(df_black$X4))
cmTglm$overall["Accuracy"]
gT <- cmTglm$overall["Accuracy"]
#k-NN classification model
Y_hatTknn <- predict(train_knn, Train1)
```

```
Y_hatTknn
cmTknn <- confusionMatrix(Y_hatTknn, factor(df_black$X4))
cmTknn$overall["Accuracy"]
kT <- cmTknn$overall["Accuracy"]
#Random Forest
Y_hatTrf <- predict(train_rf, Train1)
Y_hatTrf
cmTrf <- confusionMatrix(Y_hatTrf, factor(df_black$X4))
cmTrf$overall["Accuracy"]
rT <- cmTrf$overall["Accuracy"]
write.csv(Y_hatTrf,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/rf/Trf.csv', row.names = F)
#Ensemble model
meanYT <- (as.numeric(Y_hatTglm)-1 + as.numeric(Y_hatTknn)-1 + as.numeric(Y_hatTrf)-1)/3

meanYT[which(meanYT >= 1)] <- 1
meanYT[which(meanYT < 1)] <- 0

cmTens <- confusionMatrix(factor(meanYT), factor(df_black$X4))
cmTens$overall["Accuracy"]

eT <- cmTens$overall["Accuracy"]

write.csv(meanYT,'C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/ens/Tens.csv', row.names = F)
```

17.16 Python and R code for pressure determination

17.16.1 Python code to display images

```
import imageio
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
import pandas as pd

# Displaying seperated R, G and B channels
fig, (ax1, ax2, ax3) = plt.subplots(1, 3)
ax1.imshow(pic[ : , : , 0])
ax1.axis('off')
ax1.set_title("Red Channel")
ax2.imshow(pic[ : , : , 1])
ax2.axis('off')
ax2.set_title("Green Channel")
ax3.imshow(pic[ : , : , 2])
ax3.axis('off')
ax3.set_title("Blue Channel")


#Displaying The cut-off images
fig, (ax1, ax2, ax3) = plt.subplots(1, 3)
pic = imageio.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG2.jpg')
red_mask = pic[:, :, 0] < 170
```

```
pic[red_mask] = 0
plt.figure(figsize=(15,15))
ax1.imshow(pic)
ax1.axis('off')
ax1.set_title("Red Channel < 170", fontsize=10)

pic = imageio.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG2.jpg')
green_mask = pic[:, :, 1] < 125
pic[green_mask] = 0
plt.figure(figsize=(15,15))
ax2.imshow(pic)
ax2.axis('off')
ax2.set_title("Green Channel < 125", fontsize=10)

pic = imageio.imread('C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/RBG2.jpg')
blue_mask = pic[:, :, 2] < 80
pic[blue_mask] = 0
plt.figure(figsize=(15,15))
ax3.imshow(pic)
ax3.axis('off')
ax3.set_title("Blue Channel < 80", fontsize=10)
```

17.16.2 R code for machine learning model training

```
library(ggthemes)
library(tidyverse)
library(caret)

P <- c(0.01,0.01,3.75,3.75,7.24,7.24,10.74,10.74,14.23,14.23)
R <- c(184,178,176,170,162,161,115,131,149,148)
G <- c(150,144,126,118,100,101,64,73,80,84)
B <- c(85,81,106,98,77,82,56,66,73,81)
data <- data.frame(P=P,R=R,G=G,B=B)
write.table(data, "C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Pressure prediction plot/traindata.txt")

testset <- data.frame(P = c(0.01,0.01,3.75,7.24,7.24,10.74,14.23,14.23), R = c(194,176,164,159,160,156,142,132), G = c(156,142,112,98,101,100,85,77), B = c(92,78,75,72,68,69,66,68))
write.table(testset, "C:/Users/InvitXCY2/OneDrive - UCL/Bureau/SCO memoire/Pressure measurements/Pressure prediction plot/realdata.txt")

# Linear regression model for pressure determination (R,G,B variable)
control <- trainControl(method = "cv", number = 20, p = .9)
train_data_rf <- train(P ~ ., method = 'lm' ,data = data, trControl = control)
d_rf <- predict(train_data_rf, newdata = testset)
percent_error_rf <- sqrt((d_rf-testset$P)^2)/testset$P
mean(percent_error_rf)
sqrt(sum((d_rf - testset$P)^2/6))

# k-NN (k=3) model for pressure determination (R,G,B variable)
control <- trainControl(method = "cv", number = 20, p = .9)
```

```
train_data_rf <- train(P ~ ., method = 'knn' ,data = data,tuneGrid = data.frame(k
= c(3)), trControl = control)
d_rf <- predict(train_data_rf, newdata = testset)
percent_error_rf <- sqrt((d_rf-testset$P)^2)/testset$P
mean(percent_error_rf)
sqrt(sum((d_rf - testset$P)^2/6))

# Random forest model for pressure determination (R,G,B variable)
control <- trainControl(method = "cv", number = 20, p = .9)
train_data_rf <- train(P ~ ., method = 'rf' ,data = data,trControl = control)
ggplot(train_data_rf, highlight = T)
d_rf <- predict(train_data_rf, newdata = testset)
percent_error_rf <- sqrt((d_rf-testset$P)^2)/testset$P
mean(percent_error_rf)
sqrt(sum((d_rf - testset$P)^2/6))

# Random forest model for pressure determination (G,B variable)
control <- trainControl(method = "cv", number = 20, p = .9)
train_data_rf_GB <- train(P ~ G + B, method = 'rf' ,data = data,trControl =
control)
ggplot(train_data_rf_GB, highlight = T)
d_rf_GB <- predict(train_data_rf_GB, newdata = testset)
percent_error_rf_GB <- sqrt((d_rf_GB-testset$P)^2)/testset$P
mean(percent_error_rf_GB)
sqrt(sum((d_rf_GB - testset$P)^2/6))

# Random forest model for pressure determination (R,G variable)
control <- trainControl(method = "cv", number = 20, p = .9)
train_data_rf_GR <- train(P ~ G + R, method = 'rf' ,data = data,trControl =
control)
ggplot(train_data_rf_GR, highlight = T)
d_rf_GR <- predict(train_data_rf_GR, newdata = testset)
percent_error_rf_GR <- sqrt((d_rf_GR-testset$P)^2)/testset$P
mean(percent_error_rf_GR)
sqrt(sum((d_rf_GR - testset$P)^2/6))

# Random Forest model for pressure determination (only G variable)
control <- trainControl(method = "cv", number = 20, p = .9)
train_data_rf <- train(P ~ G, method = 'rf' ,data = data,trControl = control)
ggplot(train_data_rf, highlight = T)
d_rf <- predict(train_data_rf, newdata = testset)
percent_error_rf <- sqrt((d_rf-testset$P)^2)/testset$P
mean(percent_error_rf)
sqrt(sum((d_rf - testset$P)^2/8))

plot <- seq(65,150,0.1)
plotx <- data.frame(G = plot)
plot2 <- predict(train_data_rf, plotx)
plot3 <- data.frame(plotx, plot2)
plot3 %>% ggplot() +
  geom_path(aes(plot2,G))

write.table(plot3,      "C:/Users/InvitXCY2/OneDrive      -      UCL/Bureau/SCO
memoire/Pressure measurements/Pressure prediction plot/RF-G.txt")

#Polynomial model (degree 4) for pressure determination (only G variable)
model_polym <- lm(P ~ poly(G,4) , data = data)
```

```
d_lm <- predict(model_polym, testset)
percent_error_lm <- sqrt((d_lm-testset$P)^2)/testset$P
mean(percent_error_lm)
sqrt(sum((d_lm - testset$P)^2/6))

plot <- seq(65,150,0.1)
plotx <- data.frame(G = plot)
plot2 <- predict(model_polym, plotx)
plot3 <- data.frame(plotx, plot2)
plot3 %>% ggplot() +
  geom_path(aes(plot2,G))

write.table(plot3,      "C:/Users/InvitXCY2/OneDrive      -      UCL/Bureau/SCO
memoire/Pressure measurements/Pressure prediction plot/polyn-G.txt")
```

17.17Random Forest

Random Forest algorithms are a supervised learning method used for classification or regression problems which consists in generating multiple basic decision trees to describe the trend of a set of data. Then, for a classification problem, the output of the random forest algorithms will be the output of the majority of the generated decision trees, while for a regression problem the mean value for all decision trees will be the output. This averaging of the result of multiple decision trees helps alleviate the main problem of decision tree algorithms: overfitting to the training data. The number of decision trees, but also the number of variables chosen to generate each trees, the cross-validation method, and much more parameters can be tuned in order to obtain better predictions. [140]

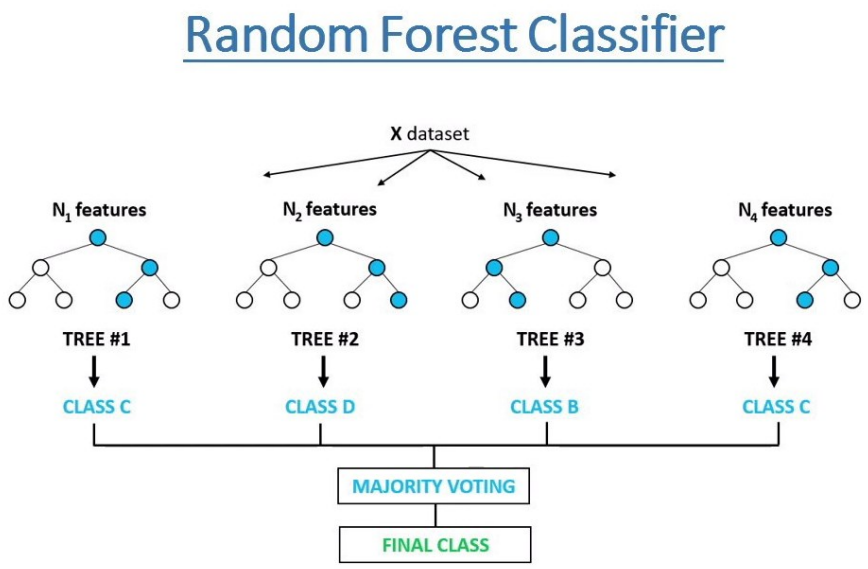


Figure 110 Example of a Random Forest algorithm with n=4 decision trees in the context of a classification problem

17.18 k-NN model

The k-NN abbreviation stands for "k-Nearest Neighbours". The k-NN algorithms also are a supervised learning method used for classification or regression problems. The principle of this algorithm is quite simple, which explains why it was one of the first machine learning algorithms developed by in 1951.[141] The output returned for a particular datapoint will be different in the classification and regression cases. For classification, the class of the datapoint will be determined by the majority class of the k-nearest points of the training dataset while in the regression case, the output corresponds to the mean value of the k-nearest points. Of course, the **k** number of nearest neighbours can be tuned for localized effects.

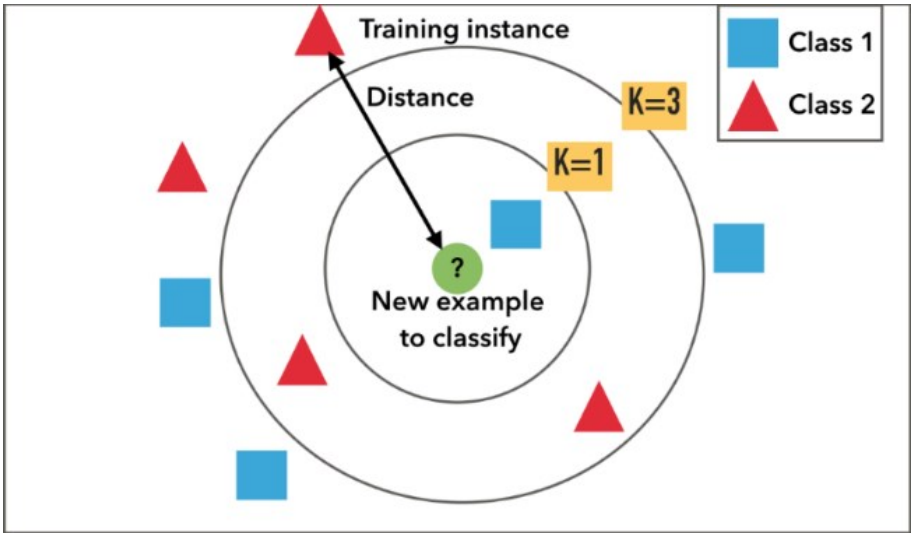
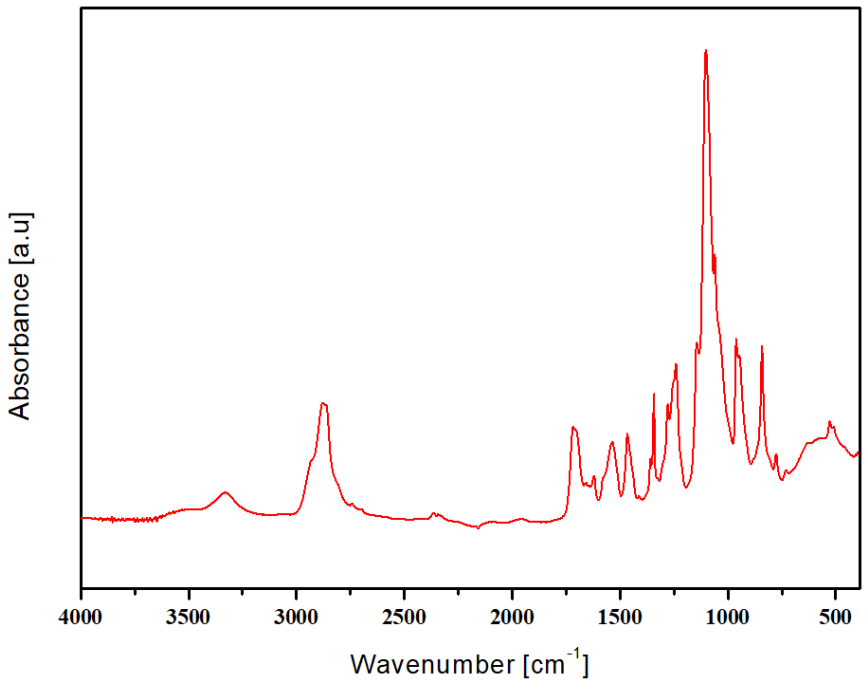


Figure 111 Example of a k-NN situation for a classification problem. For k=1, the returned class for the unknown point is 1, while for k = 3, it is 2

17.19 Fluorescein-PU FTIR spectrum



17.20 Fluorescein-PU+ $Fe(NH_2trz)_3]Br_2.nH_2O$ FTIR spectrum