Cell Reports Physical Science, Volume *1*

Supplemental Information

Ultrasensitive and Selective Detection of SARS-CoV-2 Using Thermotropic Liquid Crystals and Image-Based Machine Learning

Yang Xu, Adil M. Rather, Shuang Song, Jen-Chun Fang, Robert L. Dupont, Ufuoma I. Kara, Yun Chang, Joel A. Paulson, Rongjun Qin, Xiaoping Bao, and Xiaoguang Wang

Note S1. Adsorption of DTAB and change in liquid crystal (LC) orientation

As described in the main text, the adsorption of DTAB on the E7 surface caused a change in the E7 orientation from planar to homeotropic while the subsequent adsorption of the $ssRNA_{probe}$ returned the orientation of E7 to a planar surface orientation. The reorientation back to a planar orientation of the E7 mesogens occurs only when the DTAB is used at an appropriate concentration coinciding with the minimum surface coverage needed for a homeotropic orientation. After various trials with varying concentrations of DTAB in the system, we found that 0.5 mM DTAB is the appropriate concentration that provides the minimum surface coverage needed, allowing for the required changes in the E7 orientation. Therefore, we selected 0.5 mM DTAB as the experimental system. In contrast, at higher DTAB concentrations, 6 mM, for example, the homeotropic orientation of E7 at the aqueous–E7 interface was unaffected by the presence of the $ssDNA_{probe}$, as shown in Figure S1.

These observations of the DTAB concentration-dependent E7 optical responses can be explained with the proposition that at low concentrations of DTAB, the $ssDNA_{probe}$ molecules are capable of introducing themselves between the DTAB molecules to readily interact with the E7 surface, while at higher DTAB concentrations, the interface is highly crowded with DTAB molecules and thus the $ssDNA_{probe}$ molecules are not able to penetrate into the layer between surfactant molecules to associate with the E7 phase. As a result, at a high DTAB concentration, the $ssDNA_{probe}$ may still bind with the surface, but no E7 reorientation would be observed (Figure S1). These results lead us to conclude that the concentration of the cationic surfactant plays a vital role in the detection of ssRNA and ssDNA molecules. Hence this experiment demonstrates the presence of an optimum concentration of DTAB needed for the detection of ssRNA and ssDNA at ultralow concentrations.

Note S2. Surface coverage of DTAB at aqueous–E7 interfaces

To provide more insights into our observation of a DTAB concentration-dependent E7 response, we performed an interfacial tension measurement between the DTAB aqueous solution and E7. The aqueous–E7 interfacial tension was measured using a pendant droplet model and the average values were calculated using 10 separate measurements from each of three different droplets. For the measurement of the aqueous–E7 interfaces in a DTAB solution, a DTAB aqueous solution was placed in a quartz cell. Next, E7 was loaded into a syringe and the tip of the needle was placed under the surface of the DTAB solution. A high-resolution camera was connected with the goniometer (KRUSS DSA 100), which was used to capture the images of the droplet. Various inputs, including density, surface tension and the volume of the droplet, were provided to the built-in software, ADVANCE, to calculate the interfacial tension. The DTAB solution was prepared in a 5 mM NaCl aqueous solution having a pH between 5.5 and 6.0 to maintain the consistency of the experimental procedure.

From our interfacial tension measurements, it was found that the interfacial tension decreases significantly with an increase in the concentration of DTAB until it reached a critical micelle concentration (CMC; approximately 14 mM at 25 ^{o}C)[1] and remained nearly constant above this concentration (Figure S2). These results lead us to conclude that a partial monolayer adsorption of DTAB was achieved on the aqueous–E7 interface at 0.5 mM DTAB and that such a concentration is sufficient enough to change the E7 orientation while allowing for the facile intercalation of the $ssDNA_{probe}$ and the $ssRNA_{CoV}$/$ssDNA_{CoV}$ target with the E7 surface, which is required for detection. To support our hypothesis, we estimated the percentage of the surface covered by the DTAB at aqueous–E7 interfaces using the following equation[2]:

$$\gamma - \gamma_{\circ} = \Gamma_{\infty} RT[\, ln\,(1 - m)] - \frac{Km^2}{2} \quad \text{S.1}$$

where R is the gas constant (8.314 J mol^{-1} K^{-1}), T is the absolute temperature, m is the fraction of surface coverage, and γ and $\gamma_{\circ}$ are the interfacial tension of the aqueous–E7 interface with and without DTAB, respectively. Γ_{∞}, which represents the maximum surface concentration for DTAB, is 3.17×10^{-3} mol/cm^2.[3] For simplicity, we set the cooperativity term $K = 0$. For concentrations up to the CMC of DTAB (approximately 14 mM), the coverage of DTAB at the aqueous–E7 interfaces increases with an increase in the concentration of DTAB, as shown in Figure S2B. When the concentration of DTAB was above the CMC, however, the surface was saturated with DTAB. We notice here that at 0.5 mM DTAB, the percentage of surface coverage is estimated to be around 36%. This calculation result supports our assumption of partial surface coverage of DTAB at 0.5 mM, and suggests that a substantial amount of the surface area at the interface is an open LC surface. We comment here that this low surface coverage of DTAB plays a critical role in ultrasensitive detection of SARS-CoV-2. In addition, we have performed additional experiments to show the effect of DTAB concentration on the sensitivity of LC sensor to SARS-CoV-2 virus. As increase in the DTAB concentration at the LC surface, the threshold concentration of SARS-CoV-2 that causes perpendicular-to-planar ordering transition of the LC increases, which is caused by the strong perpendicular anchoring of the LC at the surface (Figure S2C).

Note S3. Effect of $ssRNA_{CoV}$ on the optical appearance of the E7 film

Grayscale intensity was used to quantify the brightness change of the E7 film with the addition of the $ssDNA_{probe}$ and the target $ssRNA_{CoV}$ over time as shown in the main text. Here we performed an adsorption of 3 fM $ssRNA_{CoV}$ below the threshold concentration of the detection limit. At such low concentrations no measurable change in the brightness of the E7 film was observed after 60

minutes, as shown in Figure S3. When the concentration of $ssRNA_{CoV}$ is 30 fM or above, the grayscale of the E7 was measured to decrease with an increase in the adsorption of $ssRNA_{CoV}$ at the surface, as shown in Figures 3D and 3E of the main text. Therefore, these results lead us to conclude that the detection limit of our designed E7-based system for $ssRNA_{CoV}$ is 30 fM.

Note S4. Adsorption of prehybridized $ssDNA_{probe}$ and $ssRNA_{CoV}$

As demonstrated in the main text, the successful adsorption of the $ssDNA_{probe}$ and $ssRNA_{CoV}$ was performed discretely in two different steps. The adsorption of the $ssDNA_{probe}$ and $ssRNA_{CoV}$ is based on the electrostatic interaction and hybridization of base pairs. To provide further insight, here we studied the adsorption of prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$ on the DTAB-decorated surface. Specifically, we first mixed the $ssDNA_{probe}$ and the $ssRNA_{CoV}$ in a 5 mM NaCl aqueous solution, and subsequently added the prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$ to the DTAB-decorated E7 film. We found that, over a wide concentration range (up to 100 nM), the prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$ caused no measurable changes to the optical appearance of the E7 film, as shown in Figure S4. Hence, the designed E7-based system is applicable for the detection of $ssRNA_{CoV}$ and is inept for double stranded or prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$. These results suggest the lack of interaction between the prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$ and the DTAB-decorated aqueous–E7 interface owing to the absence of electrostatic charges on the prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$.

Note S5. Development of a machine learning-based smartphone-based application (App) for detection kit

Our recognition system was built based on 88 independent sample images collected using a smartphone. 29 samples that were exposed to $ssRNA_{CoV}$ with concentrations $\geq$ 30 fM were marked

as positive, and the rest were marked as negative and categorized as either 'exposed to SARS' or 'not exposed'. The recognition system has two modules: (1) LC area detection, which locates the LC-infused specimen grid from the images, and (2) a patch-based machine learning system for classifying the texture of LC-infused grids.

Detection of the LC-infused grid

Since the LC-based detection kit presented a regular circular shape with textures, we adopted an image template matching method to locate the desired LC-infused grid. This method operated a per-pixel type of search, which scanned over the images of the detection kit, as shown in Figure S5. This method returned a location with a maximum correlation to the template.[4] Because of the images varied in brightness due to different environmental illumination conditions, the template matching needed to operate on a transformed image which was invariant to lightness and/or brightness differences. Additionally, the images might be taken at different distances to the detection kit, resulting in scale differences. To address these challenges, we used a multi-scale template matching algorithm to determine the location of the LC-infused grid:

$$\arg\min_{\Omega \in K,s} \int_{\Omega,s} |J(\boldsymbol{p}) - \Psi(I(\boldsymbol{p}))|^2 d\boldsymbol{p} \qquad \text{S.2}$$

where K is the full smartphone image space, I refers to a specific smartphone image, s refers to the scale of the template to accommodate smartphone images taken at different distance to the image, and Ω is a subset of the image space, which is parameterized by the its location and shape. J is a template of the LC-infused specimen grid at a normalized size, and Ψ refers to a brightness invariant transformation to allow the LC-infused specimen grid location algorithm to operate on images under different lighting conditions. Here we used the well-known Canny edge operator for Ψ[5]. We note here that equation (S2) is the same as equation (2) in the main text.

The optimization of equation (S2) was performed using a regular square-based template. Specifically, the distance metric of $|J(\boldsymbol{p}) - \Psi(I(\boldsymbol{p}))|^2$ was transformed to a correlation of $J(\boldsymbol{p})$ and $\Psi\big(I(\boldsymbol{p})\big)$. Figure S6 shows the matching probability map (the matching result) of six methods provided by OpenCV.[6] In our work, we selected cross correlation (shown in Figure S6C) as the matching probability estimator, and the corresponding equation can be written as:

$$CCORR(x, y) = \sum_{x',y'} (J(x', y') \cdot \Psi(I(x + x', y + y')) \quad \text{S.3}$$

where I is an input image to be detected and $\Psi(I)$ refers to the edge transformation. *CCORR* is the probability map which is built by sliding the template over the image and computing at each location. For more details about template matching, we would refer readers to Reference 4.

Next, we took a multi-scale approach for template matching, in which an image pyramid is built on the source images.[7] As shown in Figure S7, the searching analyzes the image with different scales and locates the LC-infused grid with the best correlation to the template. In our system, we defined 20 scales on the image to cover most of the distances that a user exercises for smartphone images.

Finally, we used an edge feature of the images to address the uncertainties caused by the different brightness levels of the image due to different environmental lighting conditions. The images were transformed through a Canny edge operator,[5] which executed a series of refinements on an edge magnitude map produced through a Laplacian operator. As shown in Figure S8, with two images of significantly different illuminations, the Canny operators captured the textural properties of the visual patterns of the LC-infused grid, and thus effectively improved the accuracy of the template matching. In summary, this multi-scale template matching method succeeded at all 88 independent images in our experiments.

Below is the code for the multi-scale template matching method:

```python
import cv2 as cv
import numpy as np

def roiDetection(IMAGE_PATH,TEMPLATE_PATH,canny_param1=20,canny_param2=50):
    '''''
    IMAGE_PATH: source image path, e.g. taken by mobilephone
    TEMPLATE_PATH: template image path
    canny_param1: parameter of Canny edge operator, default 20
    canny_param2: parameter of Canny edge operator, default 50

    '''
    # Load Template, convert to gray scale and apply edge operator
    template = cv.imread(TEMPLATE_PATH)
    template = cv.cvtColor(template, cv.COLOR_BGR2GRAY)
    template = cv.Canny(template, canny_param1, canny_param2)
    tH, tW = template.shape

    # Load source image, convert to gray scale
    image = cv.cvtColor(cv.imread(IMAGE_PATH), cv.COLOR_BGR2RGB)
    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

    found = None

    # Loop over the scales of the source image
    for scale in np.linspace(0.5, 2.0, 20)[::-1]:
        rH = int(gray.shape[0]*scale)
        rW = int(gray.shape[1]*scale)
        # Resize the image according to the scale
        resized = cv.resize(gray, (rH, rW))

        r = gray.shape[1] / float(resized.shape[1])

        if resized.shape[0] < tH or resized.shape[1] < tW:
            break

        # Apply edge operator on source image
        edged = cv.Canny(resized, canny_param1, canny_param2)
        # Execute template matching
        result = cv.matchTemplate(edged, template, cv.TM_CCOEFF)
        (_, maxVal, _, maxLoc) = cv.minMaxLoc(result)

        # Record location, scale, and response value of matching probablity
        if found is None or maxVal > found[0]:
            found = (maxVal, maxLoc, r)

    # Get the location and scale with maximal matching probability across all scales
    (_, maxLoc, r) = found

    # Compute corresponding bounding box at scale 1.0
    (startX, startY) = (int(maxLoc[0] * r), int(maxLoc[1] * r))
    (endX, endY) = (int((maxLoc[0] + tW) * r), int((maxLoc[1] + tH) * r))

    # Crop Region of Interest from source image
    ROI_image = image[startY:endY,startX:endX,:]
    return ROI_image, (startX,startY,endX,endY)
```

Machine learning system for visual pattern recognition of the LC-infused grid

To detect the status of the LC-infused grid in the image, we designed a feature based on the textural properties and optical appearance of the LC-infused grid. This procedure consists of three steps: (1) radiometric correction for the images taken under a variety of environmental lighting conditions, (2) color transformation from RGB to a CIELAB[8] space for color distance metric computation, and (3) feature vector extraction.

Radiometric correction

To address the challenge that the images might be taken by users under different lighting conditions, we used a radiometric correction which transforms the images to be lightness-invariant. Here we used a linear model to represent the illumination by correcting the image patches with homogenous lightness over the LC-infused grid region. The linear model consists of three parameters, which form a plane in three-dimensional space, which can be written as:

$$\hat{I}_{\mathrm{x,y}} = a \cdot x + b \cdot y + c \qquad \text{S.4}$$

where a, b, c are parameters of the model, and $\hat{I}_{\mathrm{x,y}}$ is the expected pixel value at the location (x, y). The corrected pixel values then are presented by residuals from a perfect plane:

$$e_{x,y} = I_{x,y} - \hat{I}_{\mathrm{x,y}} \qquad \text{S.5}$$

in which $I_{x,y}$ is the raw pixel measurement. A representative example of an unevenly illuminated image patch is shown in Figure S9A. Inspection of Figure S9B and S9C reveals that, compared with a simple mean correction (*i.e.*, zero-mean values), the plane-based correction recovered a well-illuminated image.

Below is the code for radiometric correction:

```python
import cv2 as cv
import numpy as np

def meanNormalize(input_image):
    '''''
    input image: a 3D numpy matrix with shape=(Height,Width,Channel)
                 and dtype=numpy.uint8
    '''
    # zero-mean the image by subtracting mean value
    return input_image.astype(np.float) - np.mean(np.mean(input_image,axis=0),axis=0)

def planeMeanNormalize(input_image):
    '''''
    input image: a 3D numpy matrix with shape=(Height,Width,Channel)
                 and dtype=numpy.uint8
    '''
    input_image = input_image.astype(np.float)
    output_img = input_image

    zn =  input_image.shape[2] # number of channels
    num_px = input_image.shape[0]*input_image.shape[1]

    # build linear system [ci, ri, 1] * [a, b, c]^T = Ii
    matA = np.zeros((num_px,3))
    matB = np.zeros(num_px)
    for zi in range(zn):
        for ri in range(input_image.shape[0]):
            for ci in range(input_image.shape[1]):
                pxidx = ri*input_image.shape[1]+ci
                matA[pxidx,0] = ci
                matA[pxidx,1] = ri
                matA[pxidx,2] = 1
                matB[pxidx] = input_image[ri,ci,zi]
        # Least Squared Solution
        x = np.linalg.lstsq(matA,matB)
        # Compute adjusted observation
        for ri in range(input_image.shape[0]):
            for ci in range(input_image.shape[1]):
                pxidx = ri*input_image.shape[1]+ci
                output_img[ri,ci,zi] = output_img[ri,ci,zi] - np.dot(x[0],matA[pxidx,:]
.ravel())
    return input_image
```

Conversion from RGB to a CIE Lab color space and feature vector extraction

We converted the RGB image to a CIE Lab color space, since it is known to be perceptually more meaningful for color distance computation. Subsequently, feature extraction was performed on the normalized image in the CIE Lab space. To homogenize the input for our features, we subdivided the ROI into 4 × 4 grids, from a resized 128 × 128 pixels image of the LC-infused grid, as shown in Figure S10. We used the standard deviation (std) metric within one grid to represent the homogeneity/heterogeneity of the LC-infused grid, which can be written as:

$$std = \sqrt{\frac{1}{|C|}\sum_{i \in C} e_i^2} \quad \text{S.6}$$

where C is the region of the grid, $|C|$ is the number of pixels in the grid, e_i is the pixel value at location i, which was corrected following the step described in section 5.2.1. The values from the grid ($4 \times 4 \times 3$, where "3" represents the number of image color channels) of std were concatenated into a 48×1 vector which served as the feature of the Support Vector Machine (SVM)[9] classifier.

Below is the code for feature vector extraction:

```
import cv2 as cv
import numpy as np

def extractFeature(image,W=4):
    '''''
    image: A 3D numpy matrix with shape=(Height,Width,Channel)
        and dtype=numpy.uint8
    W: output grid height and width, default 4
    '''
    # Resize input with any size to 128 x 128 x Channel
    resized = cv.resize(image, (128, 128))
    # Convert to CIELAB color space
    lab = cv.cvtColor(resized.astype(np.float32)/255.,cv.COLOR_RGB2LAB)
    # Radiometric correction with linear model
    normaized = planeMeanNormalize(lab)

    # Width of each cell of grid
    TW = int(128/W)
    C=normaized.shape[2]
    output_grid = np.zeros((W,W,C),dtype=np.float)

    # Loop over each grid cell and each channel
    for ci in range(W):
        for ri in range(W):
            block = normaized[ri*TW:(ri+1)*TW,ci*TW:(ci+1)*TW,:]
            for zi in range(C):
                # Compute standard derivation of pixels in
                std = np.sqrt(np.mean(block[:,:,zi].ravel()**2))
                output_grid[ri,ci,zi] = std
    # Convert WxWxC 3d grid to (WxWxC) x 1 vector
    featureVec = output_grid.ravel()
    return featureVec
```

App implementation

The Android smartphone App was built using Android Studio. Video S3 shows a typical use case of our application, including the start page and the test results. The operation of the App was

designed to be user friendly and robust. For example, the users can either take a picture with the camera of the mobile phone or select one from the photo gallery. The App can automatically detect the region where the LC-infused grid is located, perform the machine learning-based analysis, and then report the results.

We verified our LC-infused grid detection module and pattern recognition module in Python on the workstation. The detection module was mainly implemented based on the OpenCV library[6], which supports both low-level image processing tools as well as high-level algorithms. The SVM classifier was trained in Python using Scikit-Learn,[10] a well-known and open-source machine learning toolkit. The model was then serialized with the PMML package,[11] an exchange format which describes the predictive models. To port to the Android operating system, we located open-sourced libraries corresponding to those used in Python.

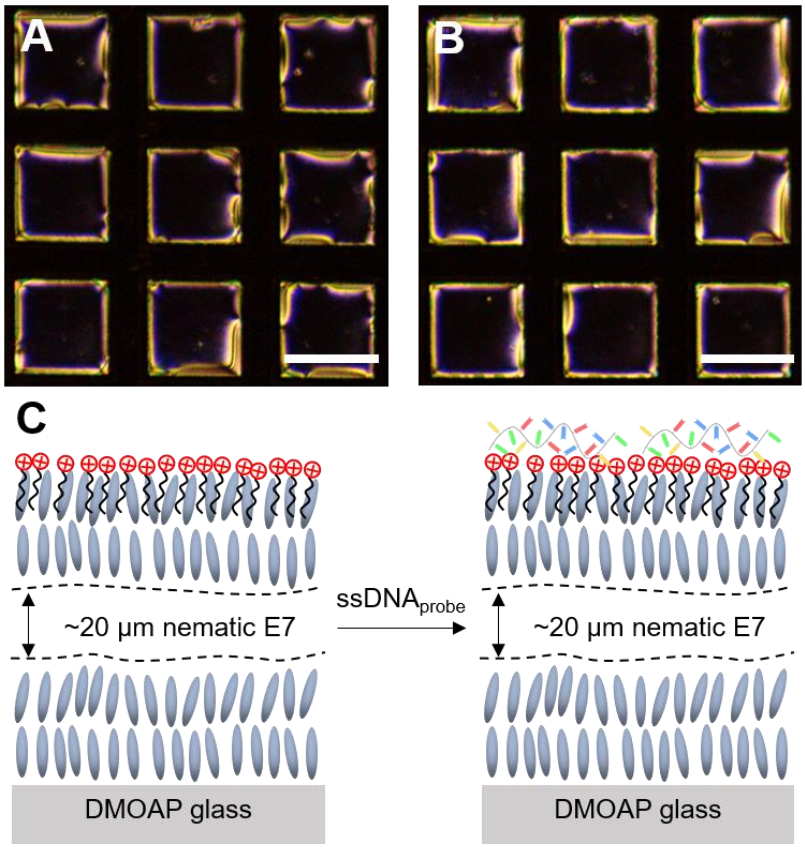

Figure S1. Adsorption of a highly concentrated solution of DTAB at the aqueous–E7 interface. (A-B) Optical micrographs (crossed polarizers) of the E7 film during (A) the adsorption of the highly concentrated solution of DTAB (6 mM) and (B) after adsorption of the ssDNA$_{probe}$. Scale bars, 100 µm. (C) Schematic illustration of the optical response of the E7 films after the adsorption of the highly concentrated solution of DTAB (6 mM) and adsorption of ssDNA$_{probe}$, respectively.

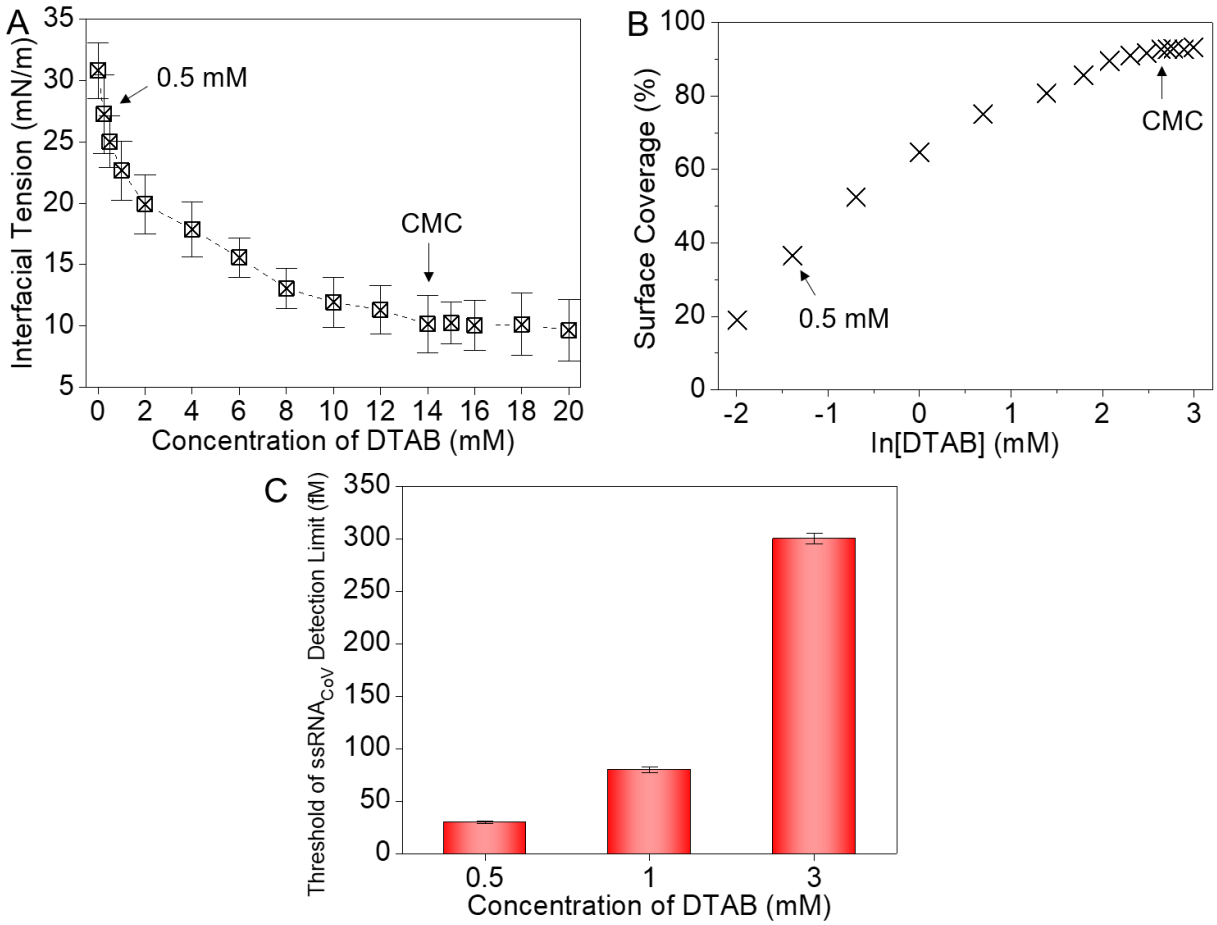

Figure S2. Surface coverage of DTAB as a function of bulk concentration. (A) Plot illustrating the change in the interfacial tension of the aqueous–E7 interface with a varying concentration of DTAB from 0 mM to 20 mM. (B) Plot of the surface coverage of DTAB at the aqueous–E7 interface as a function of the bulk concentration of DTAB in the aqueous phase. The aqueous phase contains 5 mM NaCl. The surface coverage of DTAB increases with an increase in the concentration of DTAB in the bulk aqueous phase, and remains nearly constant beyond the critical micelle concentration of DTAB. (C) Plot illustrating the threshold of SARS-CoV-2 detection limit with different DTAB concentrations. The error bars are represented as mean of three separate measurements.

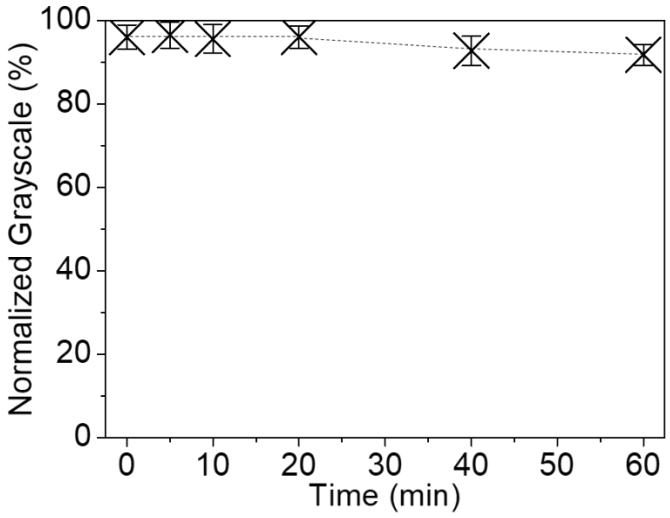

Figure S3. Plot displaying the normalized grayscale intensity of DTAB/ssDNA$_{probe}$-decorated E7 film upon adsorption of ssRNA$_{CoV}$ as a function of time. The concentration of ssRNA$_{CoV}$ is 3 fM. The error bars are represented as mean of three separate measurements.

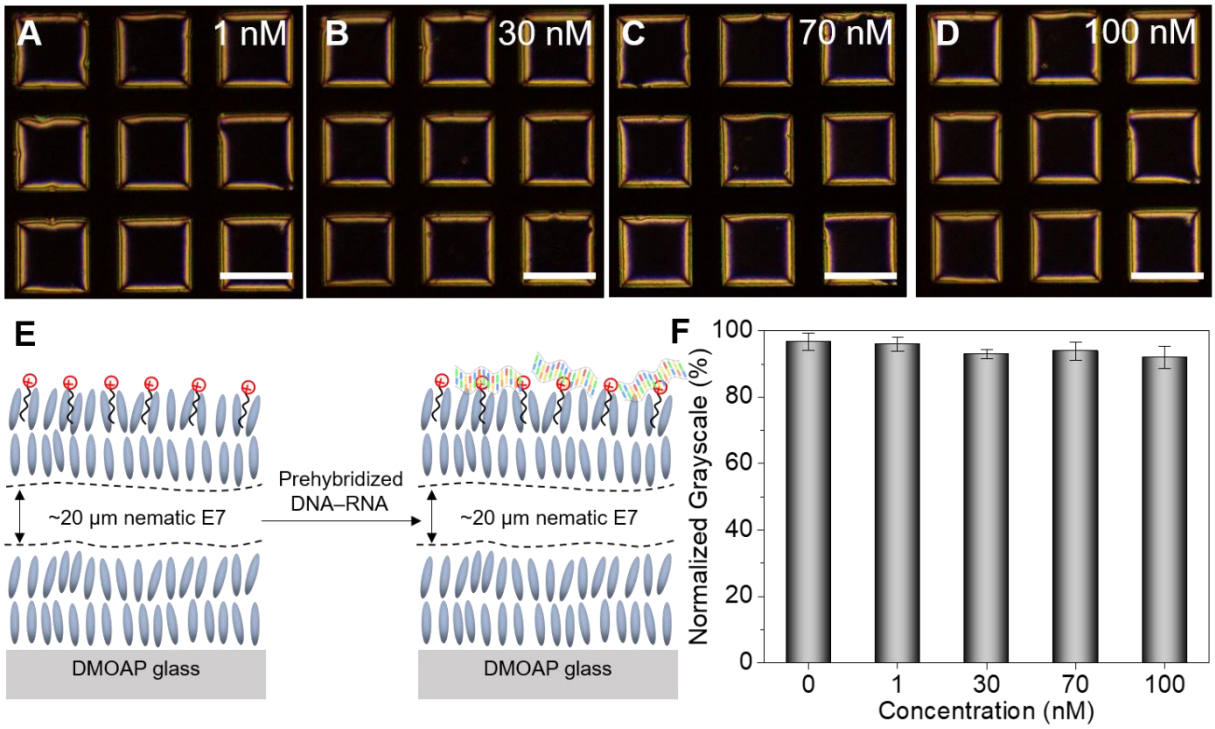

Figure S4. Adsorption of prehybridized DNA–RNA ($ssDNA_{probe}$–$ssRNA_{CoV}$) on the aqueous–E7 interface. (A-D) Optical micrographs (crossed polarizers) of the E7 film after the adsorption of prehybridized DNA with different concentrations including (A) 1 nM, (B) 30 nM, (C) 70 nM, and (D) 100 nM. Scale bars, 100 μm. (E) Schematic illustration of the optical response of the DTAB-decorated E7 film to the adsorption of prehybridized DNA. (F) Plot illustrating the normalized grayscale intensity of the DTAB-decorated E7 films with different concentrations of prehybridized $ssDNA_{probe}$–$ssRNA_{CoV}$. The error bars are represented as mean of three separate measurements.

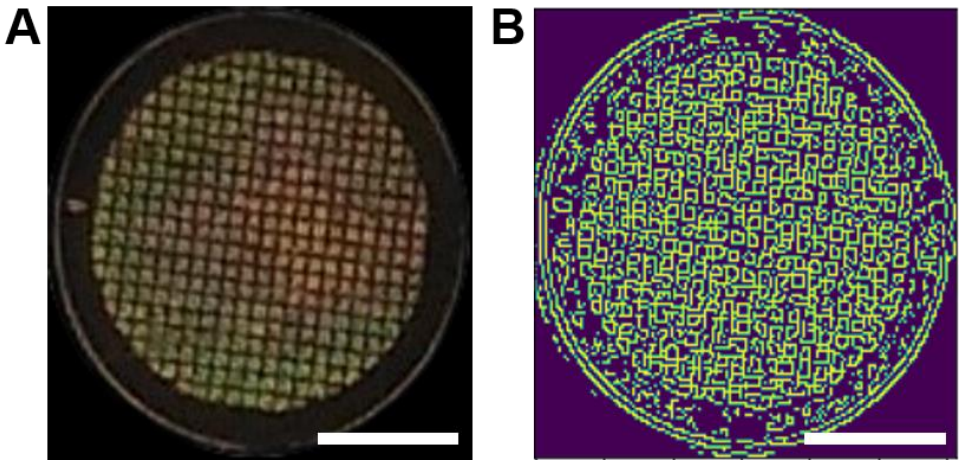

Figure S5. Image template of LC-infused grid. (A) RGB image of the template, consisting of 3 channels. (B) A single band image with the brightness invariant feature of the template extracted by Canny edge operator. Scale bars, 1 mm.

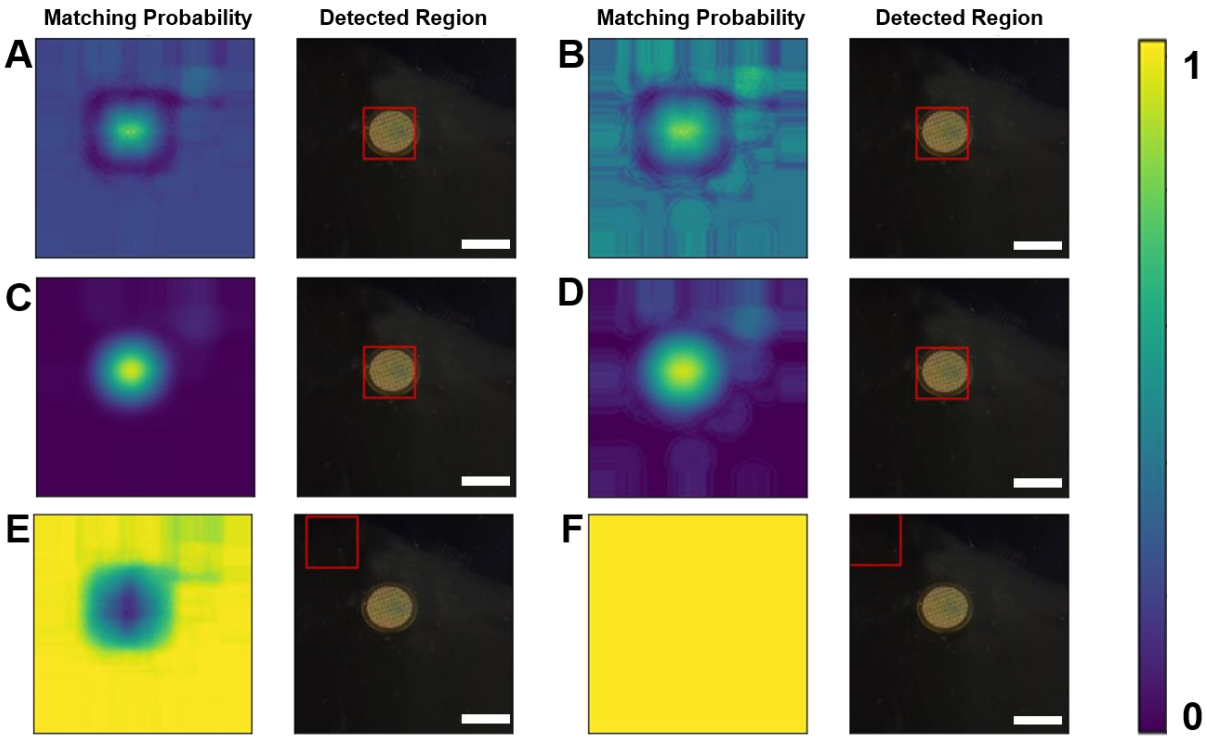

Figure S6. Performance evaluation of different template matching metrics. (A-F) Matching probability and detection regions of (A) Correlation Coefficient, (B) Normalized Correlation Coefficient, (C) Cross Correlation, (D) Normalized Cross Correlation, (E) Squared Distance and (F) Normalized Squared Distance. The red box in the right side of each panel shows the location with highest matching probability. In our work, we selected (C) Cross Correlation as the matching metric due to the best distribution of the matching probability. Scale bars, 3 mm. The color bar represents the matching probability.

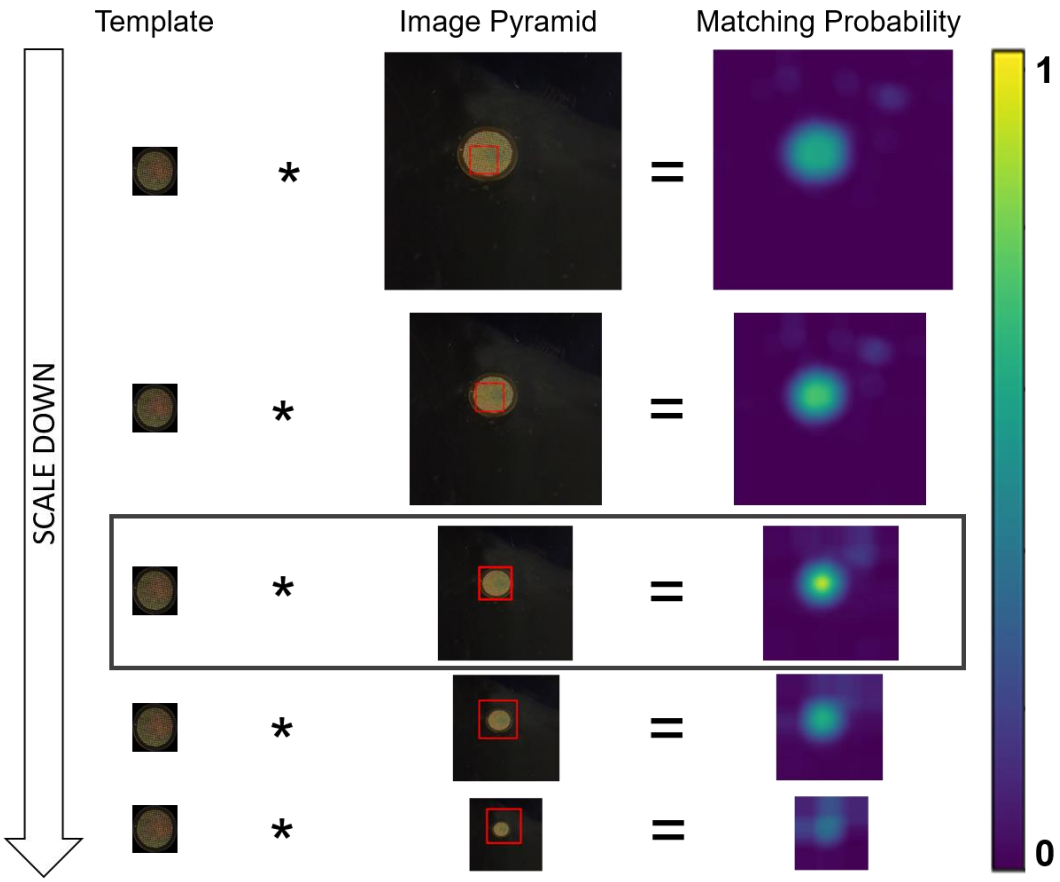

Figure S7. Multi-scale template matching with image pyramid. The size of the template is fixed while resampling input image with different scales. The red boxes mark the best solution determined by the probability value across the scale space. The color bar represents the matching probability.

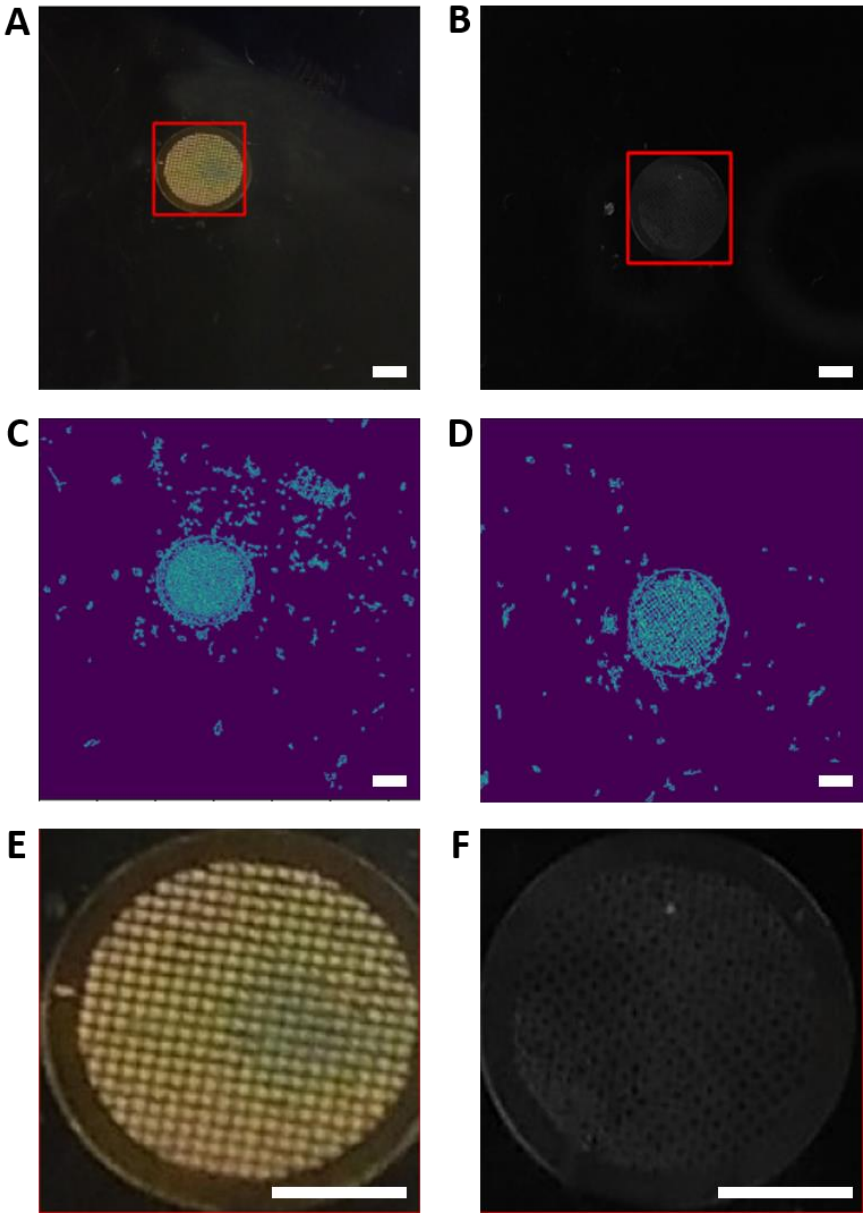

Figure S8. Brightness invariant transformation. (A-B) The image of the LC-infused grid with (A) high and (B) low brightness. Red boxes indicate the location of the LC-infused grid. (C**)** and (D) are the feature map extracted by Canny edge operator from (A) and (B), respectively. (E) and (F) show the corresponding pixel content within the red box of (A) and (B), respectively. Scale bars, 1 mm.

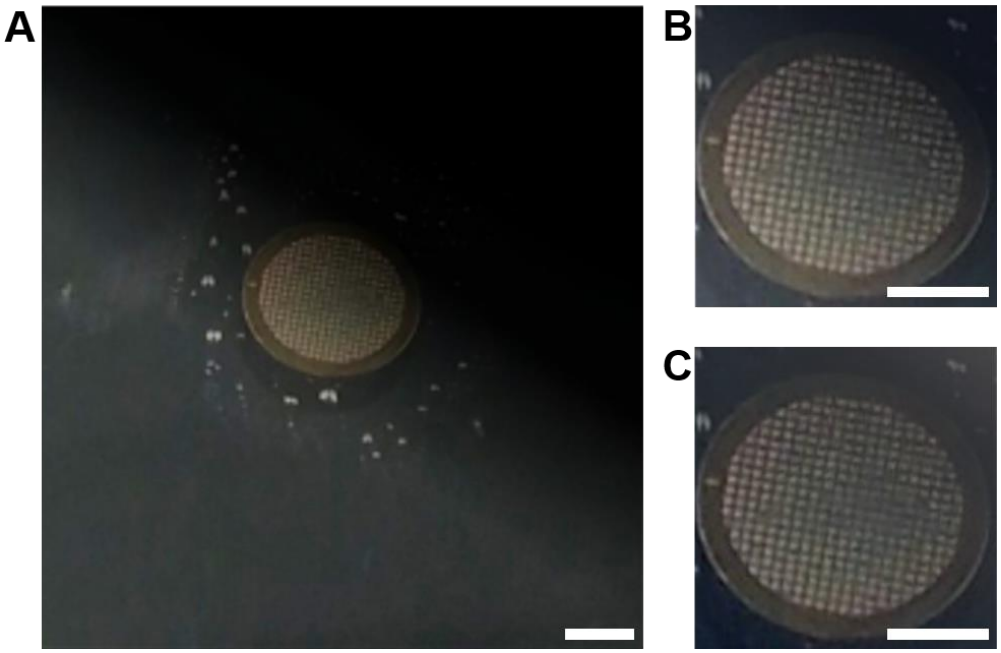

Figure S9. Uneven illumination correction. (A) A representative image of the LC-infused grid under uneven lighting condition. (B) Radiometric correction that subtracts the mean value. (C) Radiometric correction that subtracts the linear model fitted with a least squared estimator. Scale bars, 1 mm.

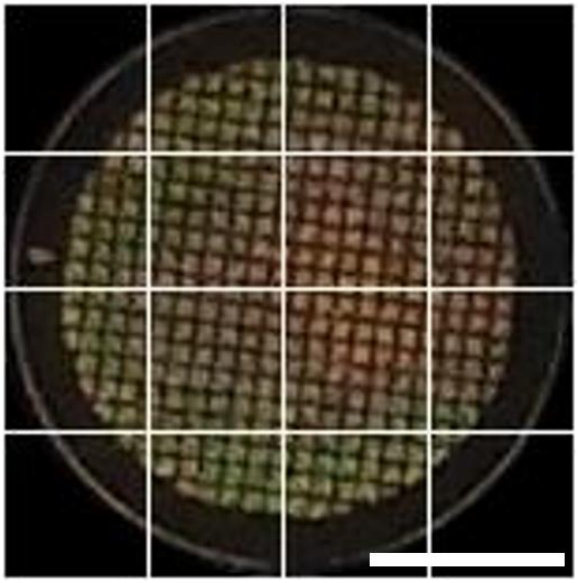

Figure S10. Spatial partition of the Region of Interest (ROI) image in feature transformation. The statistics are computed in each grid and each channel. The grid of values is then transformed to a feature vector by concatenating all values. Scale bar, 1 mm.

Supplemental References

1. Laven, J., and de With, G. (2011). Should the Gibbs analysis be revised? Langmuir *27,* 7958–7962.
2. Menger, F. M., and Rizvi, S. A. (2011). Relationship between surface tension and surface coverage. Langmuir *27,* 13975–13977.
3. Sokołowski, A., Bieniecki, A., Wilk, K. A., and Burczyk, B. (1995). Surface activity and micelle formation of chemodegradable cationic surfactants containing the 1,3-dioxolane moiety. Colloid Surface A *98,* 73–79.
4. Brunelli, R. (2009). Template Matching Techniques in Computer Vision: Theory and Practice (John Wiley & Sons, Hoboken, NJ, USA).
5. Canny, J. (1986). A computational approach to edge detection. IEEE T. Pattern Anal. *6,* 679–698.
6. Bradski, G., and Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library (O'Reilly Media, Inc).
7. Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. RCA Eng. *29,* 33–41.
8. Joblove, G. H., and Greenberg, D. (1978). Color spaces for computer graphics. SIGGRAPH Computer Graphics *12,* 20–25.
9. Chang, C.-C., and Lin, C.-J. (2011). Libsvm: a library for support vector machines. ACM T. Intell. Syst. Technol. *2,* 1–27.
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in Python. J. Mach. Learn. Res. *12,* 2825–2830.
11. Guazzelli, A., Zeller, M., Lin, W. C., and Williams, G. (2009). PMML: an open standard for sharing models. R J. *1,* 60–65.