Supplementary Materials

A Biosensor-Integrated Filtration Device for Nanoparticle Isolation and Label-Free Imaging

Leyang Liu[1,2], Takhmina Ayupova[2,3], Saurabh Umrao[2,4,5], Lucas D. Akin[2,4], Han-Keun Lee[1,2], Joseph Tibbs[2,3], Xing Wang[2,3,4,5], Utkan Demirci[6], Brian T. Cunningham[1,2,3,4,5,7] *

[1] Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[2] Nick Holonyak Jr. Micro and Nanotechnology Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[3] Department of Bioengineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[4] Department of Chemistry, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[5] Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[6] Canary Center at Stanford for Cancer Early Detection, Stanford School of Medicine, Stanford University, Palo Alto, CA, 94304, USA

[7] Cancer Center at Illinois, Urbana, IL, 61801, USA

* Author to whom correspondence should be addressed: bcunning@illinois.edu

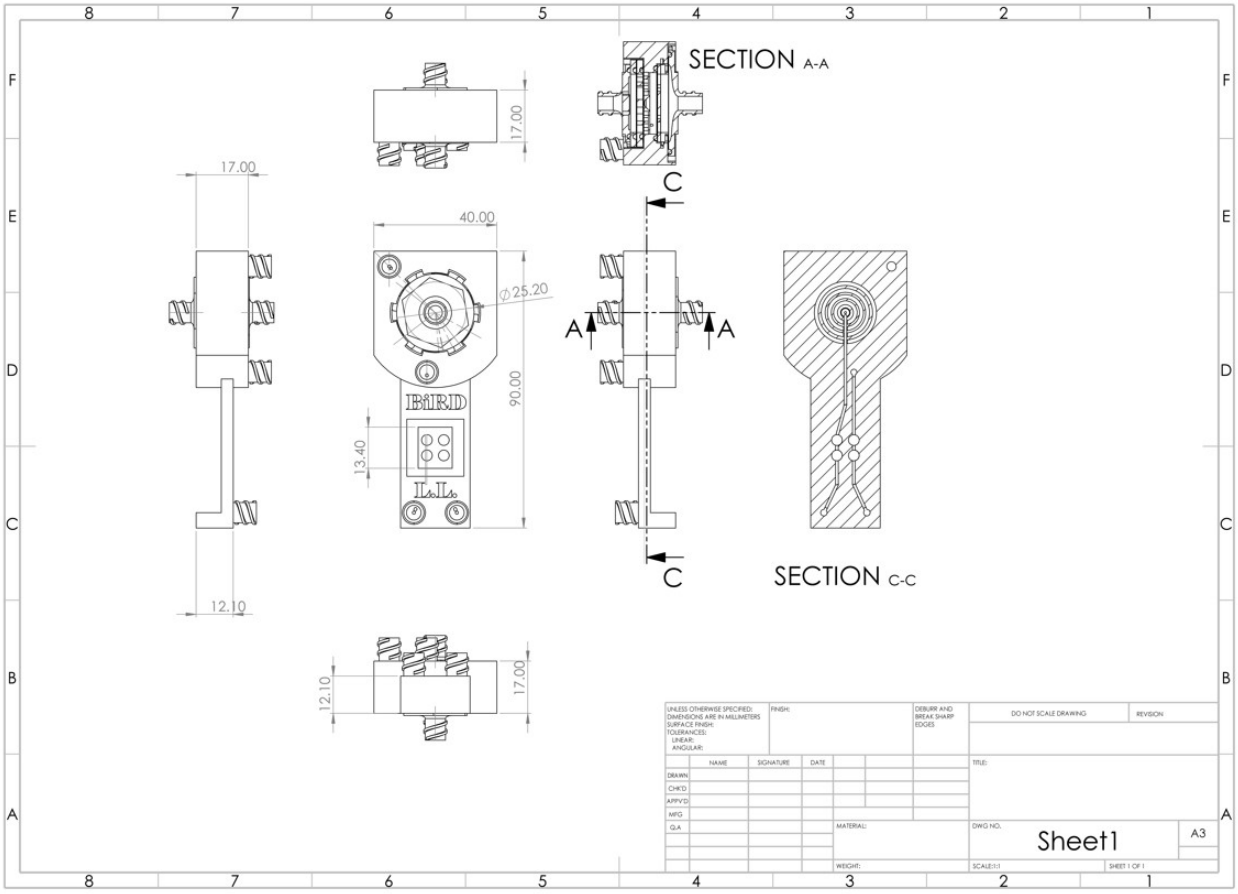

Figure S1. A drawing sheet describing the geometry of BIRD, with two cross-sectional views indicating how it is assembled and the inner channels.

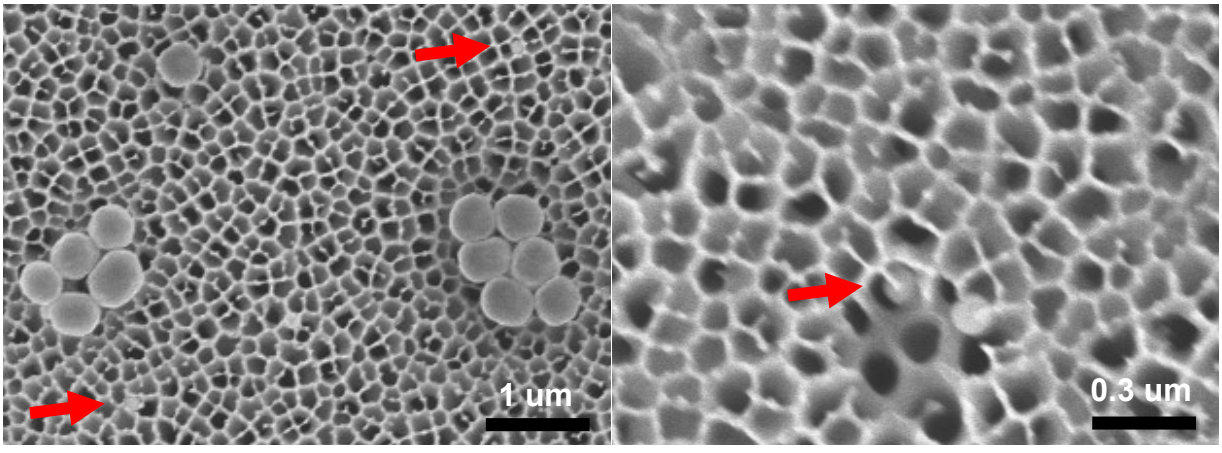

Figure S2. The red arrows indicate places where a 100 nm AuNP is observed inside the pores of 200 nm- and 100 nm-pore size AAO membrane filters.

Additional Discussion of the Selection of Membrane Filters. The most essential part of designing a filtration device is selecting the appropriate membrane filters. Many parameters must be considered, such as pore size, chemical compatibility, porosity, protein binding efficiency, and mechanical properties. Since we aim to isolate HIV from human plasma, it is critical to include a prefiltration membrane filter to remove large particles that could quickly degrade the performance of the subsequent filters. We designed experiments on how the flow rate changes under a constant pulling force to characterize the device under different filter configurations. This was done by measuring the time needed for 0.5 mL of liquid to go through BIRD. The syringe is pulled by a digital Newton meter at a force of around 5 N, corresponding to a constant applied negative pressure of 4.39 PSI. As shown in Fig. S2a, the BIRD is completely fouled after processing 4 mL of 0.1X plasma without any prefiltration, but it can filter more plasma if a 450 nm-pore size cellulose acetate (CA) prefilter is used. We tried three different types of prefiltration media, namely CA, polyethersulfone (PES) (Supor PES; Cytiva, Marlborough, MA, USA), and glass microfiber (Grade F; Sterlitech, Meredith, WA, USA). CA and PES membrane filters performed similarly to each other, while we observed a significant loss of targeting particles with the glass microfiber membrane filter.

Regarding the sieving filtration membrane filters, the most common selections are polycarbonate track-etched (PCTE) filters and anodized aluminum oxide (AAO) filters. PCTE filters are favored for their well-defined pore sizes, perfectly circular pores, and affordable prices. AAO filters are preferred for their high porosity (25% to 50%), low protein-binding efficiency, and chemical inertness. The commercially available PCTE filters can be found in many different pore sizes, but AAO filters with support rings can only be found in 20 nm-, 100 nm-, and 200 nm-pore diameters. Since we are working with human plasma saturated with all kinds of nanoparticles, we would favor a membrane filter that is less likely to clog. We performed similar flow rate tests with different AAO and PCTE filters, and the results are shown in Fig. S2b-c. The results indicate that 100 nm AAO filters are a better selection over 20 nm AAO filters for choosing Filter 3 in Fig. 1. Although 400 nm PCTE filters performed slightly better than 200 nm AAO filters (Fig. S2c), its large pore sizes are less ideal.

We note that the performance of the BIRD device is improved if the sample media is less complicated than human plasma. We conducted recovery tests on 150 nm AuNP in a buffer with BIRDs installed only with 20 nm AAO filters, and BIRD was installed with 150 nm AAO filters and CA prefilter. For simple media such as PBS buffer, where very few interferents are present, 20 nm AAO filters perform better, yielding an average recovery rate of 79.4% (Fig. S2d). This indicates that more than twenty percent of the nanoparticles are lost during the prefiltration step due to the relatively large pore size selection. If one is interested in applications such as isolating exosomes from urine, BIRD will perform better with a smaller pore-size membrane filter.

We further tested BIRD with only a 20 nm AAO filter on inactivated HIV in 1X PBS buffer, and the results suggested an average recovery rate of 77.5%. The experiment was repeated five times, and the distribution of the HIV recovery was larger than that of the 150 nm AuNP recovery due to the size uniformity of the viruses (Fig. S3a). By altering the sample-to-elution volume ratio, we could achieve enrichment of particle concentrations for both AuNP and HIV, as shown in Fig. S3b, which could be critical for biosensing at very low concentrations.

Considering all the factors and experiments mentioned above, we decided to choose a combination of 450 nm CA prefiltration membrane filter, 200 nm AAO size-exclusion membrane filter, and 100 nm AAO size-exclusion membrane filter for the application of isolating HIVs from human plasma samples.

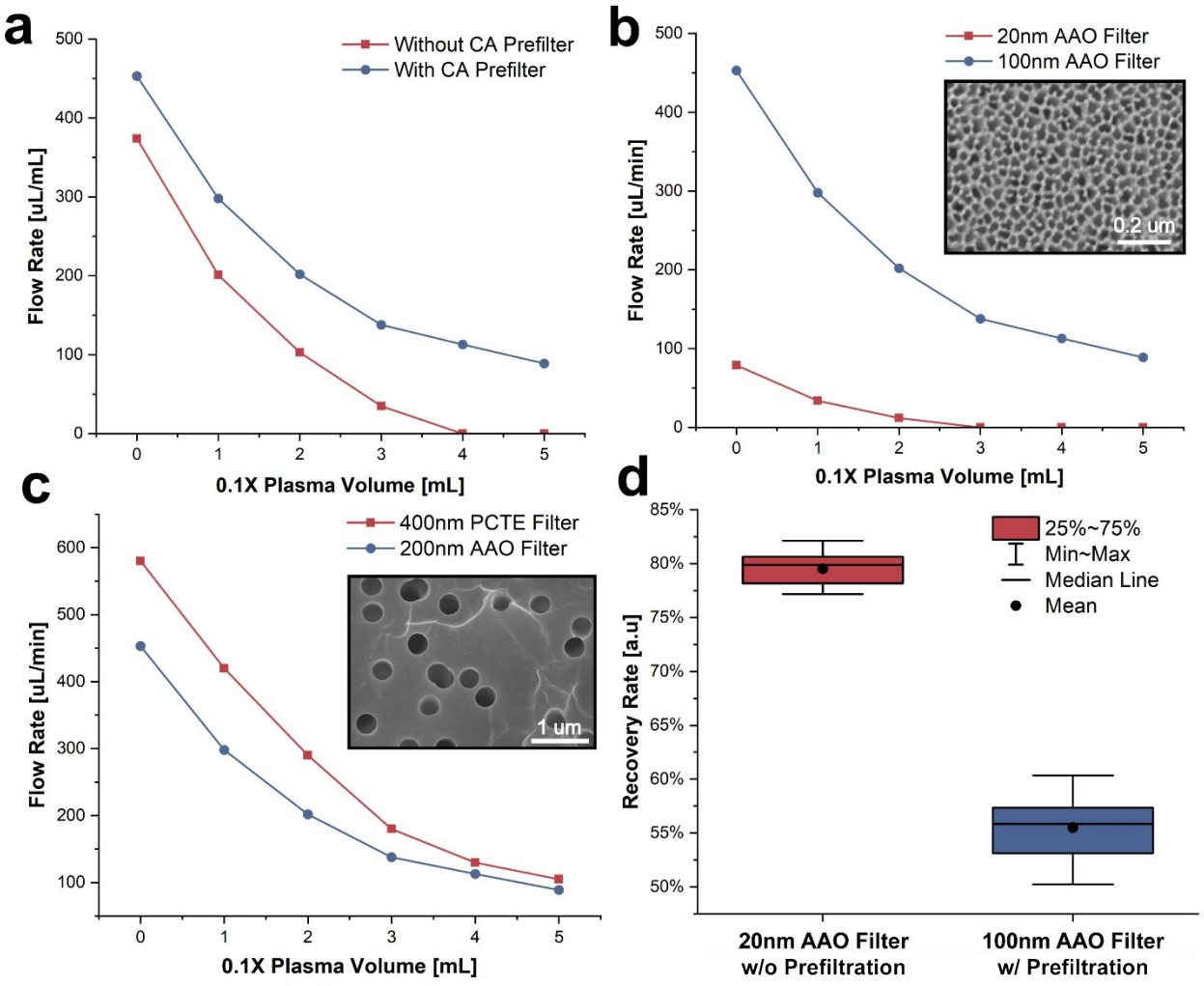

Figure S3. Filter selection experiments. (a) The change of flow rate after processing plasma with and without prefiltration. (b) The change of flow rate after processing plasma with 20 nm and 100 nm AAO filters. The inset is the SEM image of 20 nm AAO filters. (c) The change of flow rate after processing plasma with 200 nm AAO filters and 400 nm PCTE filters. The inset is the SEM image of 400 nm PCTE filters. (d) The recovery rate comparison between using a 20 nm AAO filter without prefiltration and a 100 nm AAO filter with prefiltration.

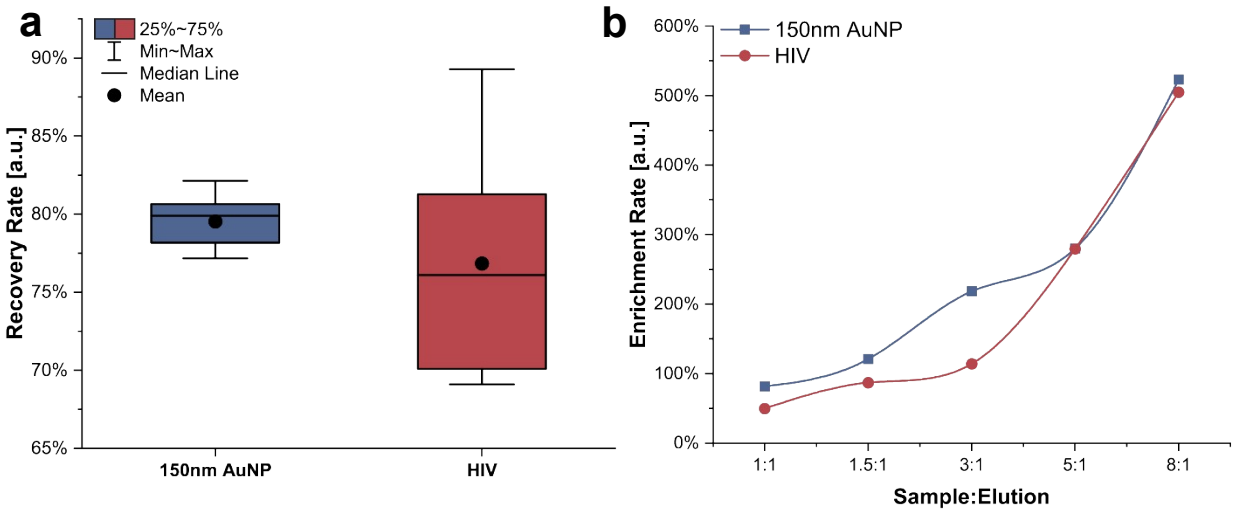

Figure S4. Recovery rate and enrichment results with only 20 nm AAO filter. (a) Recovery rate results on 150 nm AuNP and HIV in PBS buffer. (b) Enrichment rate compared to initial concentration at a different sample-to-elution volume ratio.

Figure S5. Setup for vacuum measurements.

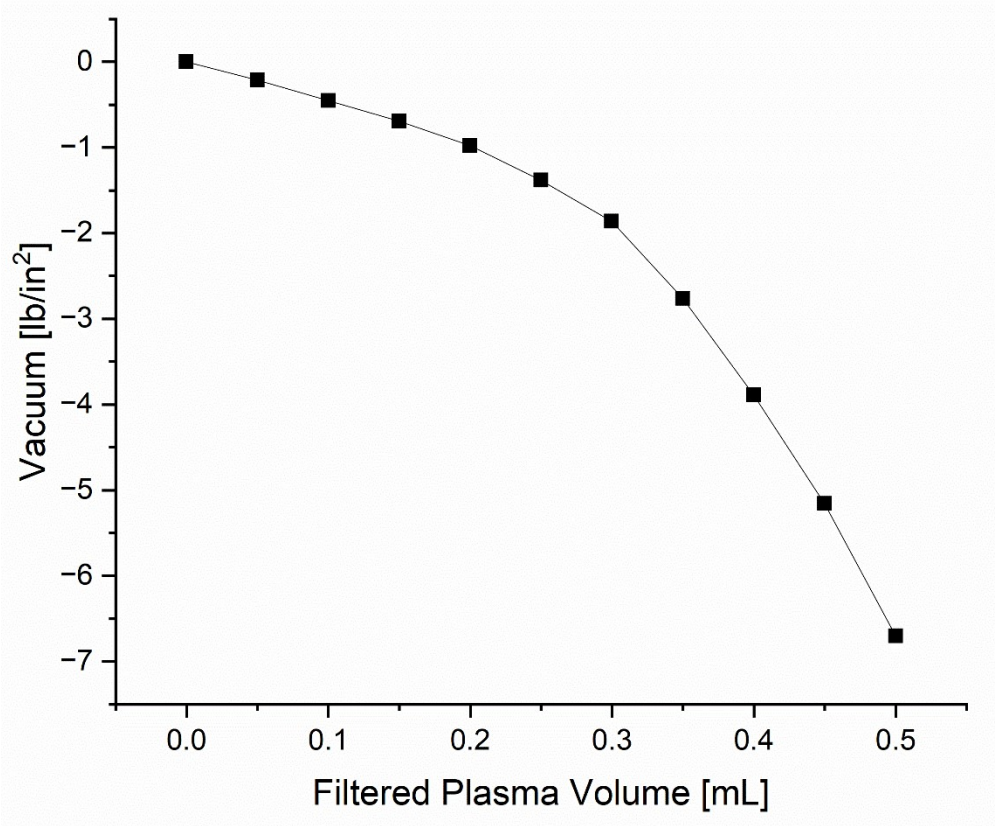

Figure S6. The vacuum pressure inside BIRD was measured after processing different amounts of undiluted human plasma. In this experiment, the sample is 0.03X diluted human plasma, and the figure is plotted with the amount before dilution.

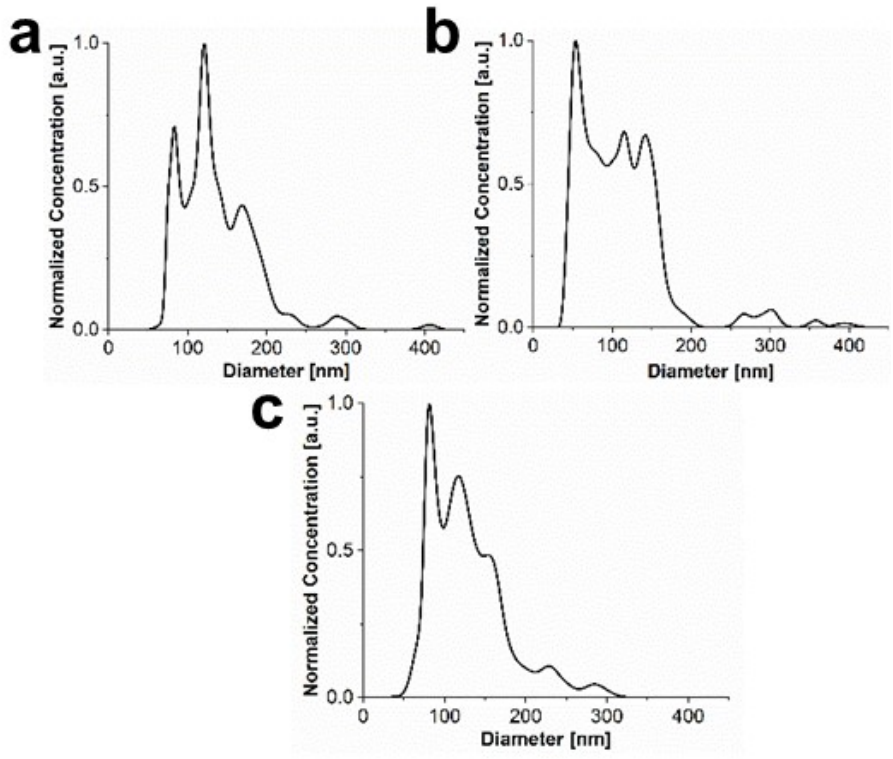

Figure S7. (a-c) NTA results for BIRD, DynaBeads Intact Virus Enrichment, and Intact Virus Precipitation Reagent, respectively.

Calculations on the number of particles per FOV. Assuming a sample with a concentration of 1e8 particles/mL and a total volume of 10 mL. The average recovery rate for AuNP and HIV in the buffer is approximately 55%, and the elution volume is 1 mL.

The eluted concentration = $\frac{1 \times 10^8 \times 0.55 \times 10}{1} = 5.5 \times 10^8\ particles/mL$.

The imaging volume can be estimated as $57.9 \times 57.9 \times 3 = 1.0057 \times 10^{-8} mL$, where the lateral dimension of the FOV is 57.9 × 57.9 μm^2. Although the field depth for a 50× objective lens (LMPLFLN50X, Olympus) is only 1.1 μm, we must account for the axial interferometric point spread function, which stretches around two μm[1]. The particle localization algorithm still recognizes the particle while it is slightly defocused. Considering all these factors, the axial dimension is set to 3 μm.

Therefore, the estimated number of particles/FOV is $5.5 \times 10^8 \times 1.0057 \times 10^{-8} = 5.53\ particles$

Image Processing Code (MATLAB).

```
N = 10;
flag_video = 1;
files = dir(“”);
sigma = 3.5;
alpha = 0.02;
folders = struct2cell(files);
folders = unique(folders(2, :));
for fold = 1:size(folders, 2)
    thisdir = folders{fold};
    filePattern = fullfile(thisdir, '*.jpg');
    files = dir(filePattern);
    imageArray = [];
    for kk = 1 : 200%size(files, 1)
        baseFileName = files(kk).name;
        tmp = (imread([files(kk).folder, '\', baseFileName]));
        imageArray(:, :, kk) = im2gray(tmp);
    end
    buf = TMF_ver1(imageArray, N);
    w = ceil(4*sigma);
    x = -w:w;
    start = 1;
    movieInfo = repmat(struct('xCoord',[],'yCoord',[],'amp',[],'sigma',[]),size(buf, 3),1);
    eMap = zeros(1024, 1024, size(buf, 3));
    for i = 1:size(buf, 3)
        buf(1, 1:4, i) = 1; %removing defect pixels of the camera
        noise = buf(:, :, i);
        if std(noise(:))<=0.02 % skipping noisy frames
            g = exp(-x.^2/(2*sigma^2));
            u = ones(1,length(x));
            img = 1-buf(:, :, i);
            imgXT = padarrayXT(img, [w w], 'symmetric');
            fg = conv2(g', g, imgXT, 'valid');
            fu = conv2(u', u, imgXT, 'valid');
            fu2 = conv2(u', u, imgXT.^2, 'valid');
            gx2 = g.*x.^2;
            imgLoG = 2*fg/sigma^2 - (conv2(g, gx2, imgXT, 'valid')+conv2(gx2, g, imgXT, 'valid'))/sigma^4;
            imgLoG = imgLoG / (2*pi*sigma^2);
            % 2-D kernel
            g = g'*g;
            n = numel(g);
            gsum = sum(g(:));
            g2sum = sum(g(:).^2);
            A_est = (fg - gsum*fu/n) / (g2sum - gsum^2/n); % estimation of amplitude
            c_est = (fu - A_est*gsum)/n;
            J = [g(:) ones(n,1)]; % g_dA g_dc
            C = inv(J'*J);
            f_c = fu2 - 2*c_est.*fu + n*c_est.^2; % f-c
            RSS = A_est.^2*g2sum - 2*A_est.*(fg - c_est*gsum) + f_c;
            sigma_e2 = RSS/(n-3);
            sigma_A = sqrt(sigma_e2*C(1,1));
            sigma_res = sqrt((RSS - (A_est*gsum+n*c_est - fu)/n)/(n-1));
            kLevel = norminv(1-alpha/2.0, 0, 1);
            SE_sigma_c = sigma_res/sqrt(2*(n-1)) * kLevel;
            df2 = (n-1) * (sigma_A.^2 + SE_sigma_c.^2).^2 ./ (sigma_A.^4 + SE_sigma_c.^4);
            scomb = sqrt((sigma_A.^2 + SE_sigma_c.^2)/n);
            T = (A_est - sigma_res*kLevel) ./ scomb;
            pval = tcdf(-T, df2);
            mask = pval <= 1E-50;
            allMax = locmax2d(imgLoG, 2*ceil(sigma)+1);
```

```
        imgLM = allMax .* mask;
        [localMaxPosX,localMaxPosY,~] = find(imgLM);
        lmIdx = sub2ind(size(imgLM), localMaxPosX, localMaxPosY);
        localMaxAmp = A_est(lmIdx);
        movieInfo(i).xCoord = [localMaxPosX zeros(size(localMaxPosX, 1), 1)];
        movieInfo(i).yCoord = [localMaxPosY zeros(size(localMaxPosX, 1), 1)];
        movieInfo(i).amp = [localMaxAmp zeros(size(localMaxPosX, 1), 1)];
        ind = sub2ind([1024, 1024, size(buf, 3)], movieInfo(i).xCoord(:, 1), movieInfo(i).yCoord(:, 1),
i*ones(size(localMaxPosX, 1), 1));
        eMap(ind) = movieInfo(i).amp(:, 1);
        if (flag_video ~= 0)
            if start == 1
                h = figure('WindowStyle','normal');
                set(gcf, 'PaperPositionMode', 'manual');
                set(gcf, 'Position', [0, 0, 1200, 400])
                v = VideoWriter([datestr(datetime('now'), 'yyyymmdd'), 'PRISM_video', '.avi']);
                v.FrameRate = 10;
                v.Quality = 100;
                open(v);
                start = 0;
            end
            subplot(131)
            imshow(buf(:, :, i), [0.95 1.05])
            title(['Raw Image f=' num2str(i)])
            subplot(132)
            imshow(-A_est, [-0.05 0.05])
            title('Processed Image')
            subplot(133)
            imshow(-A_est, [-0.05 0.05])
            title('Detected Signal')
            hold on
            plot(localMaxPosY, localMaxPosX, 'b+')
            hold off
            if i == 1
                set(gcf,'nextplot','replacechildren');
            end
            frame = getframe(h);
            writeVideo(v,frame.cdata);
        else
        end
    end
end
if (flag_video ~= 0)
    close(v);
end

PRISM_signal = eMap(eMap ~= 0);
figure
hist(PRISM_signal, 100)
xlim([0.01 0.1])
set(gca, 'XScale', 'log')
s.(['fold' mat2str(fold)]) = PRISM_signal;
end
```

Image Acquisition Code (MATLAB).

```
vid = videoinput('pointgrey', 1, 'F7_Raw8_2448x2048_Mode0');
FOV = [512 512];  % Set the region of interest (ROI)
vid.ROIPosition = [968 768 FOV(1) FOV(2)];% Center
src = getselectedsource(vid);
src.FrameRateMode = 'Auto';
```

```
src.ExposureMode = 'Manual';
src.GainMode = 'Manual';
src.ShutterMode = 'Manual';
src.Shutter = 0.007;
N = 5; % Set moving average window size (number of frames)
constrast_window = 50;
start(vid);
frameBuffer = zeros(FOV(2), FOV(1), N);  % Buffer to store N frames
fig = figure;
fig.Position = [1000,300,800,600];
stopButton = uicontrol('Style', 'pushbutton', 'String', 'Stop Capture', ...
                    'Position', [10 10 100 30], 'Callback', @(src, event) stopCapture(fig));
saveButton = uicontrol('Style','pushbutton','String','Save images','Position',[110 10 100 30],'Callback',@(src,event)
save_image(fig));
durationLabel = uicontrol('Style', 'text', 'Position', [220 10 80 30], 'String', 'Saved Duration[s]:');
durationInput = uicontrol('Style', 'edit', 'Position', [310 10 60 30], 'String', '10');  % Default to 10 second
foldernameLabel = uicontrol('Style', 'text', 'Position', [380 10 100 30], 'String', 'Folder Name:');
foldernameInput = uicontrol('Style', 'edit', 'Position', [490 10 100 30], 'String', 'video_capture');  % Default folder name
fig.UserData = struct('vid', vid, 'captureRunning', true, 'durationInput', durationInput, 'foldernameInput',
foldernameInput);
while isvalid(vid) && fig.UserData.captureRunning
    rawFrame = getsnapshot(vid);
    frameGray = reshape(rawFrame, FOV(2), FOV(1));
    frameBuffer = cat(3, frameGray, frameBuffer(:, :, 1:end-1));
    filteredFrame = double(frameBuffer(:,:,floor(N/2))) - mean(frameBuffer, 3);
    filteredFrame = uint8(filteredFrame);  % Convert back to uint8 for display
    sample_area = filteredFrame(floor(size(filteredFrame,1)/2)-
constrast_window:floor(size(filteredFrame,1)/2)+constrast_window, ...
                            floor(size(filteredFrame,2)/2)-constrast_window:floor(size(filteredFrame,2)/2)+constrast_window);
    constrast = max(sample_area(:)) - min(sample_area(:));
    subplot(1, 2, 1);
    imshow(frameGray);
    title('Original Frame');
    subplot(1, 2, 2);
    imshow(filteredFrame, [mean(filteredFrame(:)) - constrast/2, mean(filteredFrame(:)) + constrast/2], 'border', 'tight',
'InitialMagnification', 100);
    title(['Filtered Frame (N = ', num2str(N), ')']);
    pause(1 / src.FrameRate);
    drawnow;
end
```

Parts	Unit Price [$]	Quantity	Total

AAO filter	12/piece	2	24
CA filter	0.18/piece	1	0.18
O-rings	0.05/each	4	0.2
Clear resin for 3D printing	0.065/mL	121	7.87
Glass window	0.25/each	1	0.25
Adhesive	0.73/yard	0.01	negligible
Total			32.5

Table S1. The cost breakdown of the BIRD without PC biosensor

References

(1) Lee, H.; Park, H. J.; Yeon, G. J.; Kim, Z. H. Amplitude and Phase Spectra of Light Scattered from a Single Nanoparticle. *ACS Photonics* **2022**, *9* (9), 3052-3059. DOI: 10.1021/acsphotonics.2c00803.