Fast Foveating Cameras for Dense Adaptive Resolution Supplementary Material

Brevin Tilmon, Eakta Jain, Silvia Ferrari, and Sanjeev Koppal

Abstract—Traditional cameras field of view (FOV) and resolution predetermine computer vision algorithm performance. These trade-offs decide the range and performance in computer vision algorithms. We present a novel foveating camera whose viewpoint is dynamically modulated by a programmable micro-electromechanical (MEMS) mirror, resulting in a natively high-angular resolution wide-FOV camera capable of densely and simultaneously imaging multiple regions of interest in a scene. We present calibrations, novel MEMS control algorithms, a real-time prototype, and comparisons in remote eye-tracking performance against a traditional smartphone, where high-angular resolution and wide-FOV are necessary, but traditionally unavailable.

1 Foveating vs Equiangular Cameras

There are equiangular, or full frame, cameras on the market today with resolutions exceeding 60MP, such as the Sony A79 IV. Then, why should we care about carefully placing pixels when pixels are abundant? The two key benefits of our camera against a larger sensor include 1. The native angular resolution increase from imaging a MEMS mirror with a telephoto lens. and 2. Being able to modulate this higher angular resolution field of view to the same field of view as a full frame camera, which is possible using a high speed MEMS mirror. *These two key benefits allow us to use a much smaller and therefore cheaper camera sensor compared to an equivalent full frame sensor with equal angular resolution*. In this section we present a simulation demonstrating how the number of required pixels for foveating cameras is quadratically less than a camera with the same angular resolution and FOV after MEMS movement.

Consider two cameras: one is our foveating camera imaging a MEMS mirror that has angular resolution α and a total FOV ω accounting for the maximum tilt of the MEMS mirror, and the other full frame camera has the same FOV = ω and angular resolution α. ω is calculated by

$$\omega = 2tan(6^o) + FOV_{fovea} \tag{1}$$

where the MEMS mirror can tilt from $\pm 6^o$, and FOV_{fovea} is the field of view of the camera imaging the MEMS mirror (see the foveated image in Figure 1). For reference, ω is the same size as the full frame image in Figure 1.

In Figure 1, a single foveated image has the same angular resolution as the full frame image, this is only possible when the full frame camera sensor has more physical pixels on the sensor than the foveating camera sensor. Our key idea is that not all regions of images are important, and we can dynamically multiplex resolution to salient regions with our foveating camera.

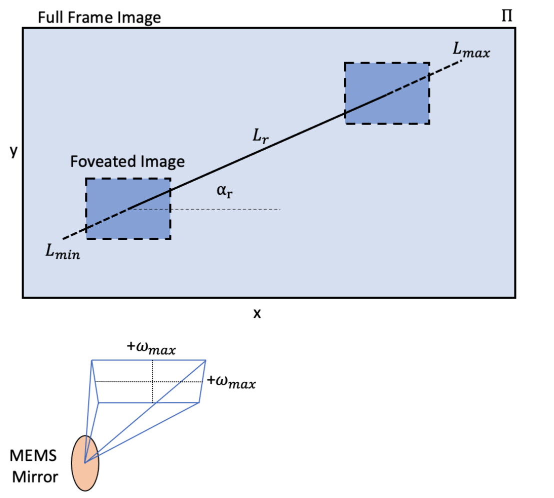

Fig. 1: Example showing multiple foveated images and a full frame camera image in virtual plane Π parallel to the MEMS mirror. A single foveated image has the same angular resolution as the full frame image. In order for the full frame camera to have the same angular resolution as the foveated camera, it must have more physical pixels on the sensor than the foveated camera.

To this end, we provide a simulation of the two camera scenario previously mentioned to demonstrate the amount of pixels needed by a full frame camera to have equal angular resolution to our foveating camera in Figure 2. We choose the popular Sony A79 IV for the foveating camera in our simulation. The Sony A79 IV has a 35mm full frame sensor with 3.76 μm pixel size, giving 60MP resolution. We use a 50mm lens as this would enable imaging the MEMS mirror at roughly 10mm from the lens. We then find the full frame resolution that gives the same angular resolution but at the maximum possible MEMS FOV determined by

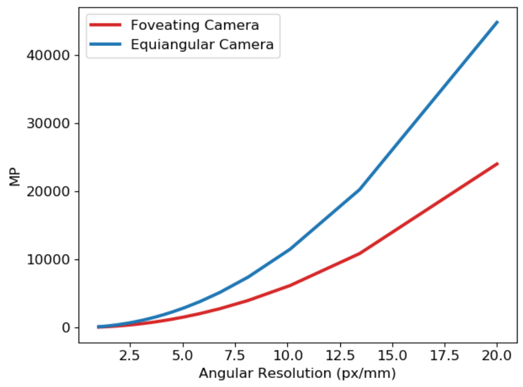

Fig. 2: As angular resolution increases, foveating cameras can obtain the same angular resolution with quadratically less pixels than their potentially gigapixel camera counterpart.

ω. Then, we shrink the physical pixel size of both cameras to simulate increased angular resolution. For context, at the baseline comparison of the foveating camera having 60MP, an equiangular or full frame camera would need 112MP. The 60MP Sony camera we began simulations with costs roughly $3,000 alongside the $1,000 MEMS mirror and driving electronics. An example 100MP camera, Fujifilm GFX100, costs $9,000. Clearly, at near gigapixel resolutions, foveating camera designs are even more pertinent due to currently high fabrication costs of large camera sensors.

2 Resolution Simulation Model

We want to vary the angular resolution of the inputs for both our foveating camera and smartphone to see which camera parameters cause the smartphone to outperform our natively higher angular resolution foveating camera. For our experiments, we scale the native pixel size and lens. We scale the pixel sizes by 0.5, 1, 1.5 and vary the foveating lens size from 2mm to 20mm. These lens sizes ensure we do not upsample above the ground truth resolution for the sensor scales since the native foveating camera lens is 35mm. Note the smartphone has smaller pixels than the foveating camera, so this gives it a denser resolution than if it had the same sized pixel as the foveating camera. We note that field of view simply measures the region sampled by the camera. The field of view can be measured in meters, degrees, radians, etc. For angular resolution, we measure the density of pixels within the sample region, typically in pixels/mm or pixels/degree.

Once the pixel size and lens are known, we simulate a new foveating camera given the original intrinsic parameters for our foveating camera. We then calculate the ratio r of pixels remaining in the original single fovea field of view, where $r < 1$. We also find the ratio for the smartphone sampling that makes the angular resolution equal to foveating camera using the smartphone intrinsic parameters. Using these ratios, we downsample the network inputs with area interpolation and then upsample back to the original resolution with bicubic interpolation to simulate a loss in angular resolution. In essence, we shrink and grow the network inputs based on how their simulated version fits in the native-constant single fovea field of view for that respective camera. We give a code example of our simulation model in Listing 1.

```

import math as m
import cv2

def simulation_model(img, lens, s_scale, camera):
    # all units in mm
    fovcam_px_size = .003 * s_scale
    iphone_px_size = 0.0015 * s_scale
    sensor = 3.28  # sensor crop both cameras
    cam_distance = 2800
    original_fov_mm = 262 # single fovea native FOV

    # 1. find new foveacam fov for new pixel size
    fov_mm = cam_distance * sensor / lens

    # 2. find new foveacam angular resolution(px/mm)
    sensor_px = sensor / fovcam_px_size
    angular_res = (sensor_px / fov_mm)

    # 3. find how much of new image fits in old fov
    new_px = original_fov_mm * angular_res

    # 4. determine how to scale original image
    if camera == 'foveacam':
        native_y = 1080
        native_x = 1920
        aspect = 1.78
    if camera == 'iphone':
        native_y = 2268
        native_x = 3024
        aspect = 1.33
    ratio_y = new_px / native_y
    ratio_x = new_px * aspect / native_x

    # 5. resize to lower angular resolution
    down = cv2.resize( img, (int(img.shape[1] *
    ratio_x), int(img.shape[0] * ratio_y)) )

    up = cv2.resize( img, (img.shape[1], img.shape
    [0]) )
    return up
```

Listing 1: Example simulation model to change angular resolution of dataset by scaling sensor pixel sizes and using different lenses.

3 Optional Kalman Filter for State and Target Tracking

A probability distribution of the targets over time is necessary to control the viewing direction of the MEMS mirror in our camera. For experiments in application section of the paper, we have used a vision-based face-tracker to estimate this distribution.

We first define a control matrix that takes the desired control vector and estimates the shift that it induces in the current state of the sensor. If we set the index for time as k, and if we denote an instantaneous time at δt, then the control matrix $B_r(k)$ for the sensor state is:

$$B_r(k) = \begin{bmatrix} \cos\alpha_r(k)\delta t & 0 \\ \sin\alpha_r(k)\delta t & 0 \\ 0 & \delta t \end{bmatrix} \mathbf{u_r} \qquad (2)$$

where infinitesimal shifts in time result in changes in the length and angle of the MEMS mirror's 1D motion. Before we write the full predict step for the filter, consider the prediction for just the MEMS mirror state, which depends on the control vector $u_r(k)$:

$$q_r(k+1) = \mathbf{I_3} q_r(k) + B_r(k) u_r(k) + Q_r, \tag{3}$$

where Q_r is the covariance matrices of the MEMS controller noise and $\mathbf{I_3}$ is the identity representing the state transition for a calibrated, controlled sensor (i.e. only our control vector and noise matters in changing the state).

To write the full **predict** step, we also add estimates of the targets. Since we are tracking two people, their 3D face locations (i.e. center of face) project onto the virtual plane Π at two 2D locations, given by $q_f = [x_{lf}\ y_{lf}\ x_{rf}\ y_{rf}]$ for the left face and right face.

We denote the full state vector as containing all the MEMS mirror information plus the face locations as $q(k) = [q_r^T(k)\ q_f^T(k)]^T$. Since we have no control over the location of the faces, the full control vector $u(k) = [u_r(k) 0]^T$.

For the target faces, the state transition vector is given calculating the optical flow of the left face $[f_{xlf}\ f_{ylf}]$ and the right face $[f_{xrf}\ f_{yrf}]$. To cast optical flow within the linear Kalman flow equations, we employ a simple trick to convert the 2D vector addition of optical flow to multiplication by setting $of_1 = \frac{x_{lf}+f_{xlf}}{x_{lf}}$, $of_2 = \frac{y_{lf}+f_{ylf}}{y_{lf}}$, $of_3 = \frac{x_{rf}+f_{xrf}}{x_{rf}}$ and $of_4 = \frac{y_{rf}+f_{yrf}}{y_{rf}}$. This allows us to specify a transition matrix $F = \left(\begin{smallmatrix} \mathbf{I_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{OF} \end{smallmatrix}\right)$, where $\mathbf{OF} = \begin{pmatrix} of_1 & 0 & 0 & 0 \\ 0 & of_2 & 0 & 0 \\ 0 & 0 & of_3 & 0 \\ 0 & 0 & 0 & of_4 \end{pmatrix}$. Augmenting $B(k) = [B_r(k); \mathbf{0}]$, we write the full predict equation as

$$q(k+1) = F\ q(k) + B(k)\ u(k) + w, \tag{4}$$

where w represents the process noise in the MEMS controller and the target motion and is denoted as covariance matrices Q_r and Q_t. Let the covariance matrix of the state vector (MEMS mirror + target faces) be $P_k = [P_r(k)\ 0; 0\ P_t(k)]$, where $P_r(k)$ is the covariance matrix representing the uncertainty in the MEMS mirror state and $P_t(k)$ is the convariance matrix representing the uncertainty in the target location. Then the change in uncertainty is

$$P(k+1) = [B_r(k)^T P_r B_r(k)\ 0; 0\ P_t] + [Q_r(k)\ 0; 0\ Q_t(k)], \tag{5}$$

where the untracked noise is represented by the covariance terms of the noise in the MEMS controller and the target Q_r and Q_t.

The **update step** for the entire system is given by two types of sensor measurements. The first is the proprioceptive sensor based on the voltage measurements made directly with a USB oscilloscope that receives the same voltages sent to the MEMS. The second is a camera that views the reflections of the mirror and applies a standard face recognition classifier to each location, determining a probability distribution of left and right face locations across the FOV. From these two measurements we can propose both the estimated state vector and its covariance matrix, $[z(k), R(k)]$. Note that the measurement function (usually denoted as $H(k)$) is the identity in our setup since all the probability distributions share the same domain, i.e. the 2D plane Π created in front of the sensor. The remaining Kalman filter equations are

$$K' = P(k+1)(P(k+1) + R(k+1))^{-1} \tag{6}$$

$$q'(k+1) = q(k+1) + K'(z(k+1) - q(k+1)) \tag{7}$$

$$P'(k+1) = P(k+1) - K' P(k+1) \tag{8}$$

3.1 Multiple Target tracking

While we focus on two target tracking in this paper, we believe it is feasible for more than two object tracking. Consider a set of gaussians $G(n) = \{g_0, g_1, .., g_N\}$ within the parallel plane Π. We propose each gaussian $\in G(n > 0)$ has a unique Lissajous descriptor $D(n) = \{[L_1, \alpha_1], [L_2, \alpha_2], .., [L_{N-1}, \alpha_{N-1}]\}$, where L is the Lissajous bound and α is the angle to a reference vector such as the the parallel lower edge of the MEMS mirror. As the gaussians move throughout Π we believe it feasible to track $G(n)$ given initial descriptors $D(n)$ by altering the Lissajous scan through traversing $G(n)$ in series and updating the next scan path according to the respective $D(n)$. Since our camera's frame rate decreased when increasing the length of $G(n)$ using the point-to-point algorithm, we assume the camera frame rate will also decrease with Lissajous scanning as $G(n)$ increases, but to a lesser extent due to Lissajous scanning speed.