EE 8374: Fundamentals of Computer Vision

## Midterm Fall 2018

## Logistics

* Files: Midterm\_EE8374.zip on Canvas.
* Assigned: October 23, 2018
* Due: November 5, 2018, 12:00 PM.

## Honor Code

The Midterm is a take-home examination. You are expected to answer each question without seeking the help or assistance of others (including but not limited to friends, colleagues or class mates). Academic dishonesty will not be tolerated. This includes cheating, plagiarism, or helping others commit a violation of the Honor Code.

Feel free to refer to the course material, online material and source code available on-line to help you in your efforts. If your work borrows from code or material available on the internet indicate that clearly in your writeup. Failure to do so will be treated as a violation of the Honor code.

Questions if any regarding the midterm should be directly addressed to the instructor ([prangara@smu.edu](mailto:prangara@smu.edu)) and not posted on Canvas. The class on Monday October 29 will be used to answer questions concerning the midterm.

## Evaluation

Performance on the midterm will be evaluated on the basis of your ability to explain your work and findings during a 10-minute one-on-one interview with the course instructor. The interview will be conducted on November 5th for in-class students. Distance students will receive additional instructions on the interview process.

## Q1: Faking miniatures [25 points]

### Overview

In recent years, a special effect called the diorama effect (or ‘miniature effect’) has gained popularity among photo-enthusiasts. The special effect derives its name from the fact that the image of real-life objects bears a close resemblance to images of miniature models. The ‘miniature effect’ is most evident when shooting distant subjects from an elevated position. An example of the effect is provided below.



Figure 1 Example of the diorama effect in photography

Image source: <http://geographical.co.uk/nature/geophoto/item/1574-a-change-of-perspective-using-tilt-and-shift-photography>

Notice that the “central-ish” rows of the image appears to be sharp and in-focus while remaining portions of the image appear blurred because of the limited depth-of-field of the camera.

As a part of this exercise, you will be developing MATLAB code to simulate the effect using digital filtering. The idea is to let the user (in this case you) define a set of rows that correspond to regions of the image that appear sharp and in-focus. We will call these set of rows the ‘Depth-of-Field’ (DOF). We are in a sense faking the optical DOF by prescribing rows in the image that the user thinks should be in-focus. The approach is only valid for camera images of distant subjects taken from an elevated position.

Rows above and below the DOF will be blurred with a Gaussian filter of increasing standard deviation (called blur scale). You are free to choose the starting and ending value of the blur scale. Reasonable values for the starting and ending blur-scales are and .

Please use the test image **Syros\_Ermoupolis.jpg** for this portion of the midterm*,* and turn in the completed MATLAB script Midterm\_FakingMiniatures.m

### Reference Material

Please read the material in these links before attempting to develop the code for faking miniatures.

* *Article:*<https://developer.apple.com/documentation/coreimage/selectively_focusing_on_an_image?language=objc>

### Details

Your implementation must support the following functionality:

1. Ability to set thefake **depth of field (DOF)** by specifying rows in the image that correspond to the regions of the scene that are in focus.
2. Ability to set the start and end blur scales and prescribe a functional form of

**Blurring**: Regions of the image further away from the DOF should have more blur. Each row .

* *Blurring color images:* Process each color channel independently. In other words, blur each color channel independently. An RGB image has three color channels so that imread returns a 3D array. Filter (using imfilter) each slice of the 3D array independently to achieve the desired effect.

Color enhancement of the final image dramatically enhances the effect of the ‘miniature effect’. Such an enhancement may be realized by converting the image from RGB format to Hue-Saturation-Value format (using rgb2hsv), scaling the saturation by an amount (>1) that produces visually compelling results in your opinion. Remember to convert the HSV image back to RGB prior to display using imshow or imagesc. The MATLAB function hsv2rgb will be useful in this regard.

* *Selecting the blur scale for each row:* Suppose the starting row of the DOF is . You could choose the blur of any row leading up-to row in a linear fashion that decreases from for the first row to for the row .

Suppose the end row of the DOF is . You could choose the blur of any row beginning with and terminating at the last row, in a linear fashion that increases from in row to for the last row.

A representative plot of the row versus the blur scale is shown below.



The aforementioned strategy for selecting the blur scale of each row is strictly linear. You are required to experiment with at-least one non-linear strategy for selecting the blur scale of each row.

A rudimentary implementation of the aforementioned approach involves blurring the entire image by a Gaussian filter with the desired blur scale and copying the intensities of the desired row into the miniature image. Extra credit will be given to implementations that can significantly speed up the process by clever optimization of the filtering process.

* *Selecting the blur kernel size:* Use the MATLAB function fspecial to generate the Gaussian filter, at a given blur scale as follows

filt\_size = 2\*ceil(3\*sigma)+1; % filter size

gaussian = fspecial('gaussian', filt\_size, sigma);

### Instructions:

Your final code should be able to, given an input image, apply blur filters such that the filtered image bears a closer resemblance to a miniature model than the original image. Run the completed script Midterm\_FakingMiniatures.m on the provided images. Include results for the provided images. Please fiddle with the parameters (start & end rows of the DOF, start and end values of the blur scale) to look for interesting outcomes.

For each result, include (a) the original input image and (b) the corresponding fake miniature.

## Q2: Creating panoramas [25 points]

### Overview

The objective of this exercise is to develop code to stitch or mosaic images of the façade of the Oxford Keble college (courtesy Oxford Visual Geometry Group) taken from three different vantage points (points of perspective). The task is to be accomplished by picking one of the three images as a reference image and attempting to align the remaining images with the reference image. The alignment requires computation of the homography or projective transformation that relates two views of the façade.

The computation of the homography begins with the identification of 4 or more matching correspondences across the two views. The SIFT detector from Assignment-4 will be used for this task.

### As discussed in class (Lecture-7) correspondences identified by direct minimization of the Euclidean distance between feature descriptors (such as vl\_ubcmatch in VL-Feat) is likely to produce erroneous matches (outliers). The problem may be addressed by using RANSAC to examine several candidate homographies and retaining the one with the most inliers. The MATLAB function estimateGeometricTransform does all of this for you. Your task is to choose the RANSAC parameters associated with estimateGeometricTransform (‘MaxNumTrials’, ‘Confidence’, ‘MaxDistance’) that work best for the task at hand.

Please use the images **keble\_a.jpg**, **keble\_b.jpg** and **keble\_c.jpg** for this portion of the midterm, and turn in the completed MATLAB script Midterm\_Panorama.m.

Please install the Computer Vision toolbox in MATLAB as you will be using functions from the toolbox to assemble the panorama.

### Details

* *Extracting keypoints & descriptors:* Extract SIFT features using the vl\_sift function in the VLFeat library.
* *Preliminary matching:* Identify putative matches or correspondences in the two views using the vl\_ubcmatch function in the VLFeat library.
* *Identify homography:* Identify the homography relating two views of the Keble college façade using the estimateGeometricTransform function in MATLAB.

### Questions

* *Show matching correspondences for each image pair (2 pairs for 3 images one of which is the reference image):* Use the MATLAB function showMatchedFeatures function with the ‘montage’ option.
* *Show correspondences after computation of the homography:* Use the MATLAB function showMatchedFeatures function with the ‘montage’ option.
* *What are the components of the* tforms *structure returned by the MATLAB function* estimateGeometricTransform*? Lookup the help associated with the use of the* tform *data structure. You may not find a MATLAB help entry for* tform*. You may instead have to lookup the help for functions that use this data structure.*
* *How is related to the homography relating the two views? What is the mathematical modeal that relates correspondences in the two views, as assumed by* estimateGeometricTransform *? (Hint: Take a look at the MATLAB help).*

### Reference Material

Please read the material in these links before attempting to develop the code for creating the panorama.

* Article: https://www.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html

WARNING: Do not replicate Step-2 of the approach described in the article. You will have to modify it to compute the homography between any image and the reference image.

## Q3: Camera Calibration [50 points]

### Overview

The objective of this exercise is three-fold:

1. Learn to calibrate a camera given multiple views of a calibration pattern (checkerboard in this case). (images courtesy University of Oulu – CMVS)
2. Measure the height of the first coin (top right) and the stick figure in the calibration image **calib0008.jpg**. Report the height in physical units.
3. Augmenting the camera view of the world by overlaying a 3D model atop the calibration checkerboard.

Please install the Computer Vision toolbox in MATLAB as you will be using apps & functions from the toolbox to answer this question.

**Step-1** relies on the assumption that the checkerboard is aligned with the plane in the world so that points on the checkerboard have . The size of each checker on the checkerboard is . Armed with this information, you can proceed to use the Camera Calibration app in MATLAB to recover the intrinsic matrix , and the pose associated with each of the checkerboard images. Use only 2 radial distortion coefficients and do not estimate skew nor tangential distortion. Export the camera calibration results to a variable (e.g. cameraParams. Save this structure to disk:

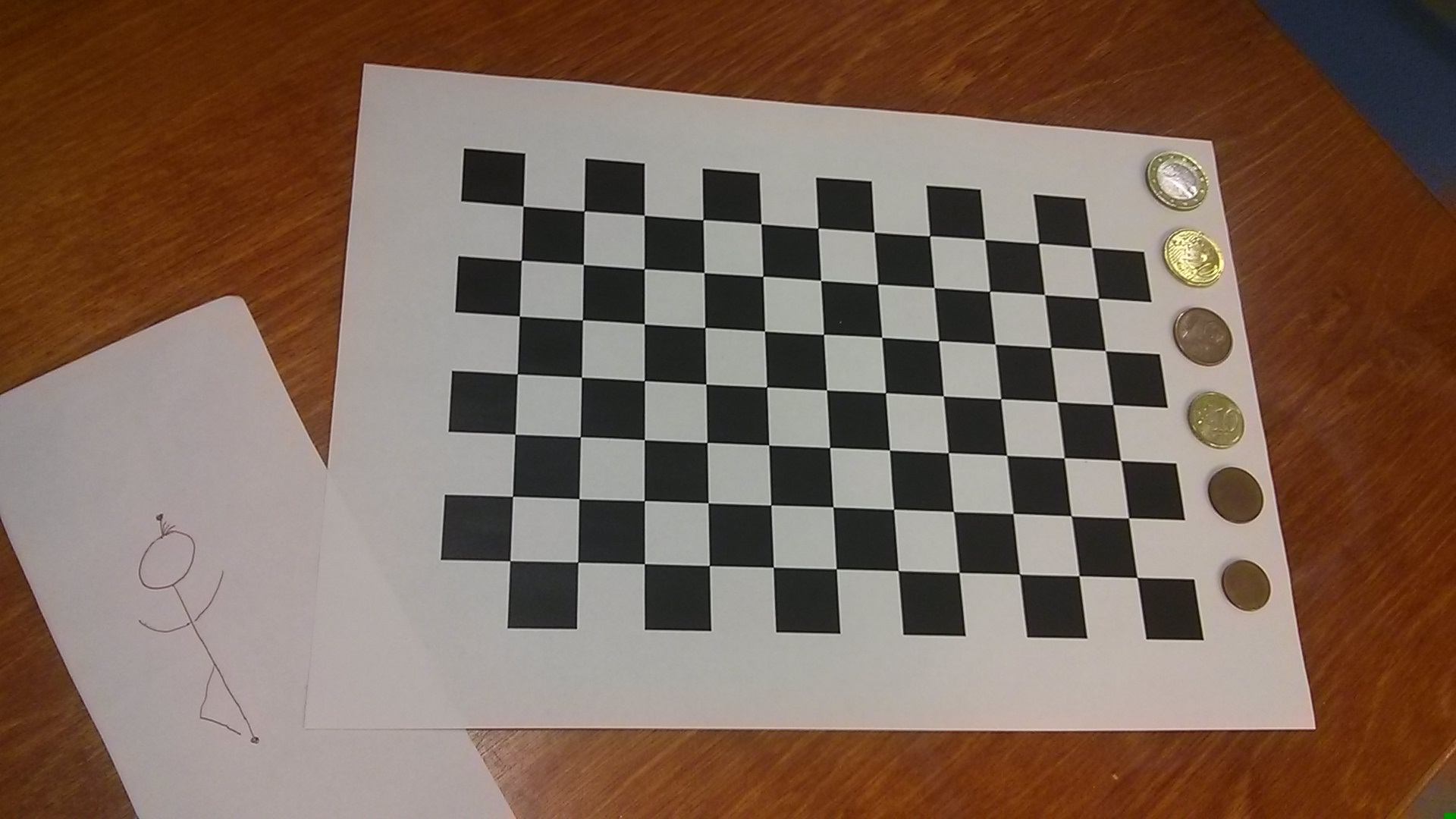
save(’calibration.mat’,’cameraParams’)

Note that the intrinsic matrix is common to all images, whereas the pose changes for each calibration image.

The MATLAB command cameraParams.RotationMatrices(:,:,8) should yield the rotation matrix associated with the 8th calibration image. In a similar fashion, you can retrieve the translation vector associated with any of the calibration images.

**Step-2** relies on inverting the projective mapping between a scene point on the plane and the corresponding image point , given by

Knowledge of the camera intrinsics and the pose derived from **Step-1** are relevant to this task. Say you wanted to identify the height of the stick figure in the following image. You begin by identifying the pixel coordinates of the top and bottom of the stick figure (identified as cyan circles in image below). Using Eq.(1) we can now attempt to recover the physical coordinates of the associated points in the physical world. Computing the Euclidean distance between these points will yield the height in millimeters.



**Step-3** relies on applying the projective mapping between a scene point in world coordinates and the corresponding image point .

Given the 3D model of a teapot, knowledge of the camera intrinsics and the camera pose , you will attemptto overlay the teapot onto the checkerboard by identifying the image coordinates of points on the teapot and displaying them on the image using the plot command in MATLAB.

Please use the following MATLAB code fragment to import the 3D model into MATLAB and adjust the 3D coordinates of the teapot, prior to overlaying it on the checkerboard.

ptCloud = pcread('teapot.ply');

XYZ = ptCloud.Location;

XYZ = XYZ \* 20;

XYZ(:,1) = XYZ(:,1) - min(XYZ(:,1)) + 20; % X-coordinate

XYZ(:,2) = XYZ(:,2) - min(XYZ(:,2)) + 20; % Y-coordinate

XYZ(:,3) = XYZ(:,3) - min(XYZ(:,3)); % Z-coordinate

The variable XYZ contains the world coordinates of points on the teapot. Use Eq.(2) to identify the corresponding image coordinates. Check if the result is consistent with the output of the MATLAB function worldToImage.

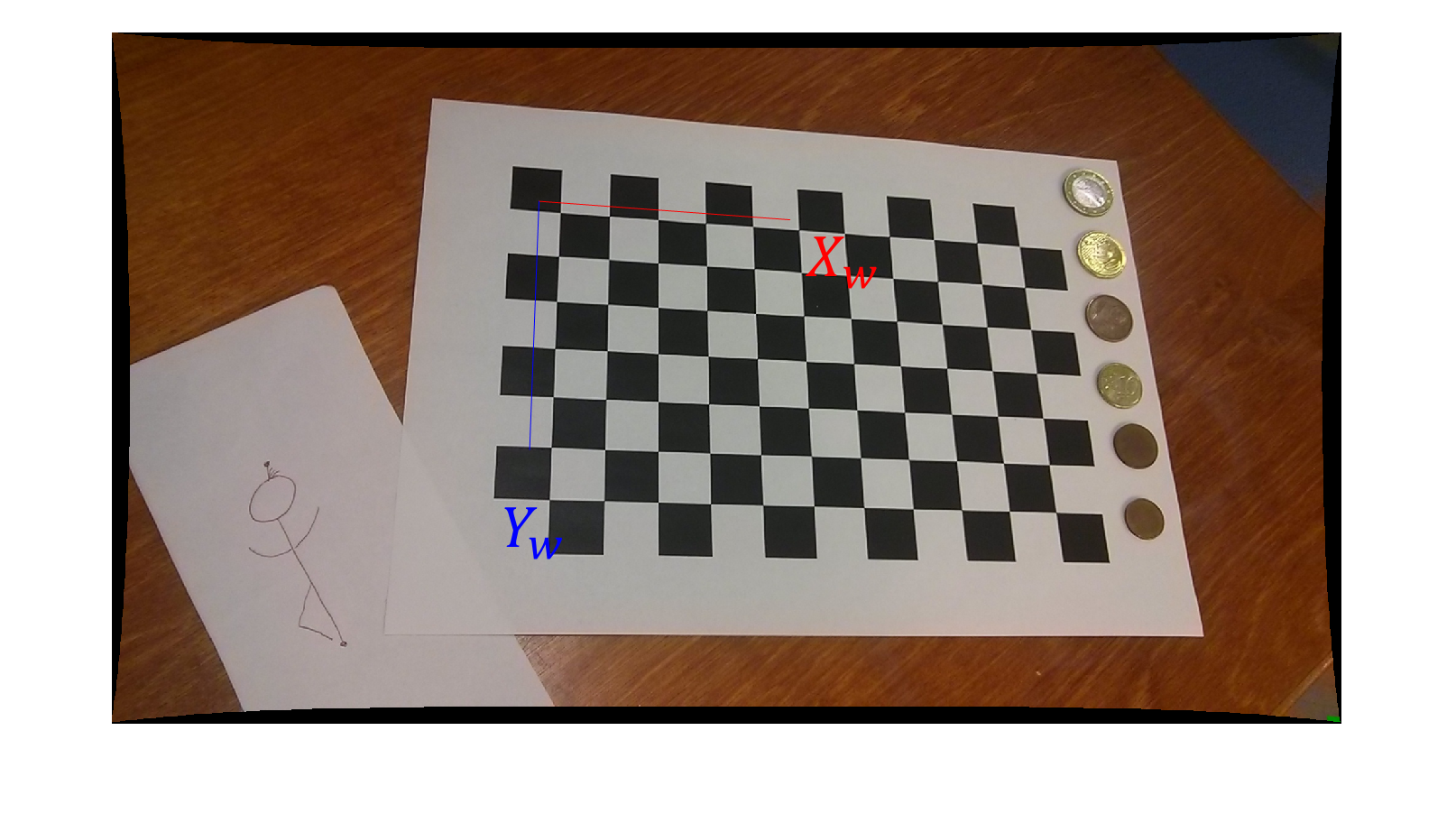
### Useful MATLAB functions: pointsToWorld, worldToImage, undistortImage

### Warnings

* Undistort the image (compensate for radial distortion) before attempting to compute the dimensions of the coin and the stick figure. Failure to do so will result in incorrect estimates of the object height. The MATLAB function undistortImage will prove useful in this regard.
* Overlay the camera image of the teapot onto the undistorted camera image of the calibration target. Failure to do so will result in incorrect rendering of the teapot. The MATLAB function undistortImage will prove useful in this regard.

### Questions

* *How and why is the assumption essential to identifying the camera intrinsics and extrinsics?*
* *What is the reprojection error and in what units is it reported? Should the reprojection error be small or large if the calibration is successful?*
* *Overlay the and axes of the world coordinate system onto the camera images of the calibration target. Here is a representative result for* **calib0008.jpg***.*



*Outline a strategy (including mathematical equations when appropriate) for overlaying the & axes of the world coordinate system. (Hint: Pick two points on the axis and find its corresponding image. The MATLAB command* pointsToWorld *will prove useful in this regard.)*

* *What is the height of the stick figure in physical units (millimeters)?*
* *What is the height in millimeters of the first coin in the top-right of* **calib0008.jpg***? Is the coin lager or smaller than a single black/white square on the checkerboard? Outline a strategy (including mathematical equations when appropriate) for identifying the size of the coin.*
* *Overlay the teapot onto the checkerboard (plane ) in each of the calibration images. Save the results and include them in your writeup.*

### Reference Material

Please read the material in these links before attempting to develop the code for this portion of the midterm.

* *Video:* [Camera Calibration with MATLAB](https://www.youtube.com/watch?v=g8SyzR0jZOA)
* *Article:* <https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html>

### Deliverables

Upload a zip file consisting of the follow three components to Canvas:

* Please name the zip file as follows <FIRST NAME>\_<LAST NAME>\_Midterm.zip
  1. Replace <FIRST NAME> with your first name
  2. Replace <LAST NAME> with your last name
* A pdf file with answers to questions posed in the midterm.
* MATLAB Source code for
  1. Midterm\_FakingMiniatures.m
  2. Midterm\_Panorama.m
  3. Midterm\_CameraCalibration.m