Assignment 1:
Curves, Surfaces & Calculus of Variations

TA in charge of the exercise: Ido Imanuel (ido.imanuel@gmail.com)
Final submission date: The 29th of November (29/11/2020)
Relevant course material: Lectures 1,2,3 & Recitations 1,2,3

1. **Homework scale:** This assignment holds **five** questions out of which you are expected to present **four**. If you would like, you may present all five for 20% extra credit. You are free to choose the questions you are most interested to solve – all hold equal weight.
2. **Submission Format.** Please submit your solutions to the appropriate homework submission box on Moodle as a ***zip*** file. Your solution zip should contain:
   - A single PDF file in with the name ***123456789_HW1_Solution.pdf*** (where the first number is your Israeli ID number, and the second number is the assignment id – 1,2,3 or 4). Your report should be **typed** and presented in **English**. No other presentation format will be accepted.
   - Source code implementation for you wet part. Please make sure the code runs directly after uncompressing the zip file. Code must be written **solely** in the Python language unless directly stated otherwise. You may use any Python module you like, but we expect to see mostly your own coding here.
   - Additional miscellaneous (such as GIF files, images etc.) as requested in the assignment.
3. **Grading.** Please clearly show your work. Partial credit will be awarded for ideas *toward* the solution, so please submit your thoughts on an exercise even if you cannot find a full solution. *If you don't know where to get started with some of these exercises, just ask!* A great way to do this is to leave questions on our Piazza forum or to schedule reception hours with the TA in charge of the exercise. Up to ten points of extra credit will be given for especially detailed solutions or show of substantial invested effort. Maximal grade in this exercise is **120**, including all possible bonuses.
4. **Collaboration and External Resources.** You are encouraged to discuss all course material with your peers, including the written and coding assignments. You are especially encouraged to seek out new friends from other disciplines (CS, Math, EE, etc.) whose experience might complement your own. However, your final work must be your own, i.e., direct collaboration on assignments is prohibited. You are allowed to refer to any external resources - if you use one, cite such help on your submission. If you are caught cheating, you will get a zero for the entire course. For now, the hand in is ***solitary*** - no couples.
5. **Late Days.** Due to the large period of time given to this exercise's submission, we will accept no late day requests unless for official Technion reasons - reserve duty, hospitalization, or tragedy. Please be responsible with this
6. **Parenting Concessions.** By Technion regulations, parents are eligible for concessions with the homework. Please contact the Head TA for details if you are eligible.
7. **Warning!** With probability 1, there are typos in this assignment. If anything in this handout does not make sense (or is blatantly wrong), let us know!

1 Triangular Mesh Framework

In this exercise you will implement basic tools for working with meshes. To evaluate your code, you are required to test it on several meshes (with and without a boundary). Please compose your results into a well-documented report where you show plots for various meshes/scalar functions. Note that you should explain every result you get, i.e., explain what you did and what can be seen in the figure so that you report is self-contained. The report should contain demonstrations of all the tasks. Handle odd end cases ***as you would like*** here.

Computational Note: You should avoid loops wherever possible. While it might seem not important at first, working later with high resolution meshes will become impossible when using a non-optimized code. Luckily, `numpy` supports vector and tensor operations natively.

Objectives:

1. **Basic Importer:** .off files are a standard format for 3D objects, along with .obj, .ply, .stl and .wrl. Write two small functions `read_off` and `write_off` that allow for reading and writing off files. The file specification may be found here[1]. You may test your loader and all other functionality with the off files attached to Recitation 1. The functions should return or receive (respectively) a tuple `(v,f)`, for the shared vertex data structure presented in Recitation 1.
2. **Basic Mesh Class:** Create a class `Mesh` whose constructor receives an .off file path string, loads the relevant mesh with your importer, placing it under `Mesh.v`,`Mesh.f` for the vertex array and the triangular face array respectively.
3. **Mesh Topology Functions:**
   a. Add to the class a function `vertex_face_adjacency()` that returns the sparse Boolean vertex-face adjacency matrix of size $|\mathcal{V}|\times|\mathcal{F}|$ defined by $A_{ij} = \text{True} \Leftrightarrow vertex\ i \sim face\ j$, where $\mathcal{V}$ denotes the vertex set, $\mathcal{F}$ denotes the face set and the symbol $\sim$ denotes a first order adjacency. Use the `scipy.sparse` package for the matrix representation.
   b. Add to the class a function `vertex_vertex_adjacency()` that returns the sparse Boolean vertex-vertex adjacency matrix of size $|\mathcal{V}|\times|\mathcal{V}|$ defined by $A_{ij} = \text{True} \Leftrightarrow vertex\ i \sim vertex\ j$.
   Think about how you can use matrix multiplication and the vertex face adjacency matrix for this computation. Remember that the diagonal should be all zeros (a vertex should not have a connection to itself).
   c. Add to the class a function `vertex_degree()` that returns a `numpy` vector of size $|\mathcal{V}|$ holding the vertex degree per vertex. How can you use the previous function to efficiently compute this?
4. **Mesh Visualization Functions:** (use the Python package **`pyvista`** as a graphical engine)
   - `render_wireframe()` - Should render the mesh in a wireframe view.
   - `render_pointcloud(scalar_func,…)` – Should render only the vertices of the mesh as spheres and color each sphere by the scalar function $f_{|\mathcal{V}|\times 1}$ which supplies a value for each vertex. Test this function by using the X,Y,Z coordinates of `Mesh.v` as scalar functions, or the vertex degree computed above. Add in additional input arguments as you see fit (colormap options for example).

[1] https://shape.cs.princeton.edu/benchmark/documentation/off_format.html

- `render_surface(scalar_func,…)` – Which should render both faces and edges of the mesh and color each face by the scalar function $f_{|\mathcal{F}|\times 1}$ supplying a value for each face or a color interpolation of $f_{|\mathcal{V}|\times 1}$ which supplies a value for each vertex. Invent relevant scalar functions as you see fit (try using random values at first, or a step function). Add in additional input arguments as you see fit (colormap options for example).

5. **Mesh Properties:** Add in functions to compute:
   - **Face Normals** – The face normals may be computed by the cross-product operator –

     $$\forall \boldsymbol{f} \in \mathcal{F} : \boldsymbol{n}_f = \boldsymbol{e}_1 \times \boldsymbol{e}_2$$

     $$\bar{\boldsymbol{n}}_f \triangleq \frac{\boldsymbol{n}_f}{\left\| \boldsymbol{n}_f \right\|}$$

     Where $\boldsymbol{e}_1, \boldsymbol{e}_2$ are edge vectors on some triangular face, received by subtraction of the two relevant vertex coordinates that each edge connects. Remember to add in an option to normalize the face normals to unit L2 norm (via a flag for example). Lastly, the normals should all point towards the "outside" of the surface.
   - **Face Barycenters** – a mean of the 3 vertices that compose the face, for each face.
   - **Face Areas** – Simply the areas of every triangular face (again use geometric identities).
   - **Barycentric Vertex Areas** – One might define an area of a vertex – see extract here[2], with numerous ways to do this. Here you will compute the simplest one – defined one third of all immediately adjacent triangle areas. How can you use the vertex-face-adjacency matrix to compute this efficiently?
   - **Vertex Normals** – Computed as a weighted by area sum over adjacent face normals:

     $$\forall \boldsymbol{v} \in \mathcal{V} : \ \boldsymbol{n}_v = \sum_{f \sim v} \mathcal{A}_f \bar{\boldsymbol{n}}_f$$

     Where the area of the face $\boldsymbol{f}$ is marked by $\mathcal{A}_f$. Remember to add in an option to normalize the vertex normals to unit norm (via a flag for example). How can you use the vertex-face-adjacency matrix and the face normals to compute this efficiently?
   - **Gaussian Curvature –** Compute the discrete Gaussian curvature for closed meshes via the Angle Defect formulation - the Gaussian curvature at each discrete vertex $\boldsymbol{v}$ is defined as 2π minus the sum of all face angles adjacent to the vertex (see figure below) normalized by the area of the vertex. We may check our numeric values of the Gaussian Curvature are correct simply by using the discrete Gauss Bonnet Theorem[3] for genus 0 meshes with no boundary:

     $$\sum_v \mathcal{K}_v \cdot \mathcal{A}_v = 2\pi\chi \ ,$$

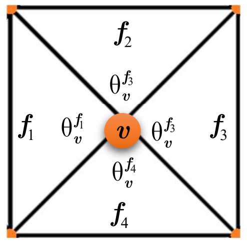
     where $\mathcal{K}_v$ is the discrete gaussian curvature at every vertex, $\mathcal{A}_v$ the relevant vertex area and χ is the Euler characteristic[4]. For most of the meshes (genus 0 meshes, with no boundary) you were given, you'll receive $\chi \equiv 2$. When displaying this scalar function, the colors might easily saturate at outliers (numerical spikes), so adjust your color limits accordingly.

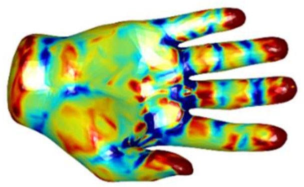
[2] https://www.alecjacobson.com/weblog/?p=1146

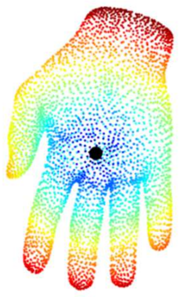
[3] https://en.wikipedia.org/wiki/Gauss%E2%80%93Bonnet_theorem

[4] https://en.wikipedia.org/wiki/Euler_characteristic

$$\mathcal{K}_v \equiv \frac{2\pi - \sum_{f\sim v} \theta_v^f}{\mathcal{A}_v}$$





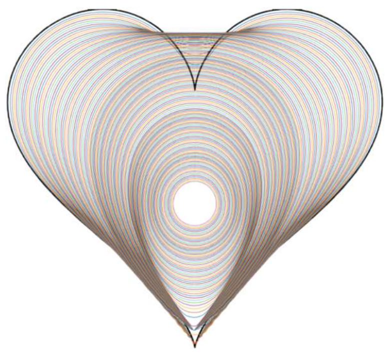


6. **Additional Visualizations**
   - Insert a function to visualize the **normalized** and **unnormalized** vertex and face normals. Use `add_arrows` or `glyph` in `pyvista`. Visualize for a few meshes. The vertex normals' arrow start point should be the vertices themselves, while for face normals – the face barycenters.
   - Visualize the areas of each face and each vertex as a scalar function via `plot_surface`.
   - Visualize the **norms** of the **unnormalized** vertex and face normals as scalar functions. Additionally, plot the respective unnormalized normals on top of the scalar function to double check everything makes sense visually.
   - Compute the vertex centroid, defined as the mean of all mesh vertices. Plot it as a sphere in 3D space, and plot with it all vertices colored by their Euclidean distance from the centroid.
7. Qualitatively explain in writing the geometric intuition behind the Angle Defect formula presented above.

2 Curve Evolution & Curve Signatures

Let $C(t)=(x(t),y(t))$ be a planar curve.

1.
   (a) Use the attached extract[5] or any other source and pick a few curves, indicating their parameterization. Approximate the curve using enough sample points and draw them using Python using a polyline discretization. **Note:** Use non-trivial curves both open and closed, and also allow for curves with various types of singularities.
   (b) Implement the geometric heat flow $C_t = \kappa\vec{N}$ (mean curvature flow) and draw evolution curves in several different times. Describe the effects of your implementation after several iterations (is it stable? Or does it break after a point? Show plots of the evolution just before and after such an event if applicable). Explain your thoughts and conclusions about the flow's behavior. Read about and relate the results to the Gage-Hamilton-Grayson theorem. Compute the arclength at each iteration and show a graph of its progression in time. For open curves, fix both end points to address boundary conditions.
   (c) **Bonus (5 Points):** Try various heuristics to smooth your progressions (overcome numerical instabilities). Bonus points will be awarded for an exceptional success for curves with singularities.



2. Demonstrate the fundamental theorem of the local theory of planar curves. Choose numerous sufficiently smooth non-trivial closed curves with enough points. Compute $\kappa(s)$, transform the curve by some Euclidean transformation (rotation, translation or reflection) and resample it to alter its parameterization. Recompute $\kappa(s)$ and show the two values are equivalent up to some initial starting point of measurement (make sure to change the starting point). Can you suggest ways to align the two signatures? Display your work and the signature before and after transformation. Report thoughts, conclusions, and difficulties.
3. Demonstrate Cartan's theorem. Choose numerous sufficiently smooth non-trivial closed curves with enough points. Compute $(\kappa(s),\kappa'(s))$, transform the curve by some Euclidean transformation and resample it to alter its parameterization. Recompute $(\kappa(s),\kappa'(s))$ and show the two values are equivalent regardless of the starting point of measurement (make sure to change the base point itself). Make sure to display the signature on the 2D plane before and after transformation. Report thoughts, conclusions, and difficulties.

[5] https://elepa.files.wordpress.com/2013/11/fifty-famous-curves.pdf

3 Image Evolution & The Heat Equation

(a) In this question we will examine the general linear heat equation in $\mathbb{R}^n$:

$$u_t = \Delta u = u_{x_1x_1} + u_{x_2x_2} + \ldots + u_{x_nx_n}$$

As a linear, continuous-time, time-invariant system, the heat equation solution may be described as a convolution integral:

$$u(x,t) = (h * f)(x,t) = \int_{\mathbb{R}^n} h(x-y,t) f(y) dy,$$

where f is some initial value $f(x)$, $x \in \mathbb{R}^n$ and $h(\cdot)$ is the system's kernel which is the system's response to an impulse function.

Find an explicit form of the heat kernel in $\mathbb{R}^n$. Show all steps.

(b) Apply the 2D linear heat equation $u_t = u_{xx} + u_{yy}$ to some gray level image $u[x,y,0] = I[x,y]$.

Program ***both***:

- The explicit form with forward Euler numerical iterations in time and central derivative approximation in space. Denote $U^n_{i,j} = u[i,j,n]$. The iterative update scheme (gradient descent process) is given formally by:

$$U^{n+1}_{i,j} = U^n_{i,j} + dt\left(\frac{U^n_{i+1,j} - 2U^n_{i,j} + U^n_{i-1,j}}{dx^2} + \frac{U^n_{i,j+1} - 2U^n_{i,j} + U^n_{i,j-1}}{dy^2}\right)$$

- A direct convolution of the image with the heat kernel itself (alternatively, multiply the Fourier transformed image with the Fourier transformed kernel).

  Attach an animation (GIF format) of the progression of both methods holding two windows: one of the image itself as a function of time, and the other of the image as a surface for the ***cameraman*** image (google for it). Detail difficulties, numerical troubles, and conclusions in verbose.

(c) Explain the connection between Gaussian smoothing, the Heat Equation and the Dirichlet energy functional:

$$E(u) = \frac{1}{2}\int_{\Omega} \|\nabla u(x)\|^2 dx$$

4 Curves

1. Let $C(t)$ be a smooth regular curve with some arbitrary parameterization in $\mathbb{R}^2$.
   (a) Prove the following formula for the signed Euclidean curvature $\kappa(t)$ of $C(t)$ with a general parameterization:
   $$\kappa(t) = \frac{\left(C'(t), C''(t)\right)}{\|C'(t)\|^3} \qquad (v_1, v_2) \triangleq \det(v_1, v_2) = \det\left(\begin{bmatrix} - & v_1 & - \\ - & v_2 & - \end{bmatrix}\right)$$
   (b) Find its explicit form in terms of the component functions $x(t), y(t)$.
2. Let $C(t)$ be a ***closed*** planar curve parameterized by $C(t) = (x(t), y(t)) \in \mathbb{R}^2$ where $t \in [0,1]$, and $g(x,y)$ be a scalar weight function defined on $\mathbb{R}^2$. Consider the following functionals:
   $$L_1(C) = \int_t |C'(t)|^2 dt \qquad L_2(C) = \int_t |C'(t)| dt \qquad L_3(C) = \int_t |C'(t)| g(C(t)) dt$$
   (a) Write down the Euler Lagrange equations for L_1, L_2 and L_3.
   (b) Relate between the Euler Lagrange equations of L_1, L_2 and L_3. Express the Euler Lagrange equations in terms of the curvature κ and the unit normal vector $\vec{N}$ at a given point on the curve.
   (c) Which functionals amongst the three are invariant to a change of parametrization?
   (d) The optical path length (OPL) is defined as the product of the Euclidean length of the path of light C, and the index of refraction of the medium through which it propagates (denoted by $n(x,y)$)
   $$OPL(C) = \int_C n(C(s)) ds,$$
   where s denotes the arclength parameterization. Consider two isotropic media (such as water, glass or air) given in the domains $\Omega \triangleq \{(x,y) \in \mathbb{R}^2 \mid y < 0\}$ and $\Omega^c \triangleq \mathbb{R}^2 \setminus \Omega$ and a scalar piecewise constant weight function defined in $\mathbb{R}^2$: $n(x,y) = \begin{cases} n_1 & (x,y) \in \Omega \\ n_2 & (x,y) \in \Omega^c \end{cases}$ where $n_1, n_2 \geq 1$ are the relevant indices of refraction. Find a connection between Snell's law, L_3, the optical path length and the Eikonal equation.

5 Surfaces

1. Consider the map $\mathcal{S}(u,v):\mathbb{R}^2 \mapsto \mathbb{R}^3$ defined by:

$$\mathcal{S}(u,v) \triangleq \frac{1}{u^2+v^2+1}\left(2u, 2v, u^2+v^2-1\right)$$

   (a) Verify that for all points $(u,v)\in\mathbb{R}^2$, $\mathcal{S}(u,v)$ lies on the unit sphere centered at the origin.

   (b) Plot this surface using `pyvista`.

   (c) Compute the Gauss Map of the surface

   (c) Compute the First and Second fundamental forms.

   (d) Compute the Gaussian and Mean curvatures.

2. Consider the map $\mathcal{S}(u,v):\mathbb{R}^2 \mapsto \mathbb{R}^3$. Show that:

$$d\mathcal{A} = \left\|\mathcal{S}_u \times \mathcal{S}_v\right\| dudv = \sqrt{\det(G)}\,dudv$$

   i.e., show that the area "stretching factor" (differential area element) as we go from the plane to the surface is given by the square root of the determinant of the metric.

3. Consider a smooth surface $\mathcal{S}$ and a point $p\in\mathcal{S}$. We would like to approximate the surface around p using a second order Taylor approximation of the form:

$$\mathcal{S}_p(x,y) \approx a + bx + cy + dxy + ex^2 + fy^2$$

   (a) Choose a coordinate system so that the point p is at the origin. What constraints does this place on $\{a,b,c,d,e,f\}$?

   (b) Find the Mean ($\mathcal{H}$) and Gaussian ($\mathcal{K}$) curvatures of the second order surface approximation (use (a)).

   (c) Prove that it is possible to choose the coordinate system so $\tilde{\mathcal{S}}(u,v) = \mathcal{S}(x(u), y(v)) \equiv gu^2 + hv^2$. What are κ_1, κ_2 in terms of g, h? What is the geometric meaning of this fact?

   (d) The sign of the Gaussian curvature is a geometric invariant. Explain in detail how would the surface approximation look like for $\mathcal{K}>0, \mathcal{K}<0$ and $\mathcal{K}=0$. Address the principal curvatures in your analysis.