MODELLING EMBRYONIC CLEAVAGE PATTERNS

Dmitry Ershov and Nicolas Minc*

Affiliation:

Institut Jacques Monod, CNRS UMR7592 and Université Paris Diderot, 15 rue Hélène Brion, 75205 Paris Cedex 13, France

*Correspondence to: nicolas.minc@ijm.fr (N.M)

Abstract:

The division patterns of early invertebrate and vertebrate embryos is key to the specification of cell fates and embryo body axes. We here describe a generic computational modelling method to quantitatively test mechanisms which specify successive division position and orientation of eggs and early blastomeres in 3D. This approach should serve to motivate and guide future experimental work on the mechanisms controlling early embryo morphogenesis.

Keywords: Early embryos; Division positioning; Microtubules; Asters; Cell Shape; Maternal domain; Yolk

1- Introduction

Early embryo development of both invertebrate and vertebrate species begin with the cleavage period. During cleavage, the egg undergoes a stereotyped choreography of subsequent reductive divisions that transform the one-cell egg into a multicellular embryo [1]. Cleavage patterns are not random, but usually follow a well ordered set of specific position and orientation of blastomere divisions. Those patterns are most often conserved among large groups of species, such as in amphibians, fishes and echinoderms, and serve as prime examples of developmental robustness. Mechanisms which control cleavage geometries remain however poorly understood. This is in part because this period is driven by maternal material in the embryo, and thus poorly amenable to genetic screens or manipulations [2]. In addition thorough live-imaging of large eggs and blastomeres implicate sophisticated methodologies [3], or may even be impaired given the opacity of many embryos. Factors which have been directly and indirectly implicated in the regulation of division plane positioning during cleavage, include cell shapes [4-7], yolk layers [8,9] and maternal cortical polarity domains [10-13]. Those may affect the organization or dynamic properties of astral microtubules which exert forces and torques to position and orient asters and mitotic spindles that specify division planes.

Recent experimental work in echinoderms, fish and frog embryos, have suggested that division plane specification in large eggs and blastomeres is instructed from interphase/ anaphase Microtubule (MT) asters which fill the large cellular volume, and probe cell shape through length-dependent MT forces [6,7]. This mechanism aligns the division axis with the cell shape long axis, and has been proposed to serve as a default cue guiding division plane orientation and position in early embryos [14,15]. Given the presence of asymmetric divisions in some blastomeres of many embryos, cell shape effect may be overridden or biased by additional cues such as vegetal polarity domains in echinoderms or by yolk layers in fish and amphibians, which alter the distribution of MT forces around cells.

We have recently implemented 3D computational models to predict division position/orientation as a function of various inputs [16]. Those models compute in a systematic manner the forces and torques exerted by astral MTs on centrosome pairs at the center of asters and output a mechanical equilibrium which correspond to the preferred division position and orientation. Those models are highly valuable to dissect the basics of cleavage patterns in many embryos. In here, we report on a set of scripts and methodologies to implement this modelling approach to test the contribution of various cues to the division patterns of early embryos.

2- Materials

Computer equipped with Mathworks Matlab (Version 2015a and above) and Surface Evolver [17]. There is no special requirements for computers, but it is recommended for better performance to have a fast CPU. On a good system a simulation of 10000 iterations with calculation of a torque map (see here after) typically takes 5-10 minutes.

Programs may be downloaded from:

DPP:	http://www.minclab.fr/wp-content/uploads/2018/01/DPP-Package.zip
Surface Evolver :	http://facstaff.susqu.edu/brakke/evolver/evolver.html

3- Methods

3.1. Generating 3D cell shapes

The initial input for the program is the 3D shape of a cell or a group of cells in an embryo, for instance, defined by the 3D coordinates of cell surfaces. This serves as a spatial reference to run a prediction on the position/orientation of the site of cell division. 3D shapes may be artificially generated using the surface minimization software Surface Evolver [17], or from a 3D segmented stack from labeled embryos imaged with a confocal or a 2-photon microscope (Fig. 1).

3.1.1 Generating an input 3D shape with Surface Evolver

We here provide some initial guidelines to generate representative 3D shapes that closely resemble real physical egg and blastomere shapes with the software Surface Evolver (SE). This software iteratively finds the surface of minimal energy under given constraints (Fig. 1A-1B). Constrains include final volume, surface tension values and confinement. The language of SE is rich and for details on structures, constrains, macros or function definition, we refer the readers to its web page.

SE requires input files that contain information on the topology and constraints of the studied surfaces. The topology is described by a simple mesh (Fig. 1A), elements of which are named and organized according to the rules of SE. The mesh describing topology is defined by vertices and their coordinates, oriented edges linking 2 vertices, oriented faces consisting of 3 edges and bodies consisting of N faces. The constraints describe the behavior of the surface; e.g. coordinates can be restricted to an arbitrary space (like a cylinder) or additional energy may be assigned to arbitrary faces, to mimic effects of differential surface tension around adhering blastomeres for instance [16]. We provide several input files in the folder DPP package\DPP files\generating_shapes_from_SE\generic shapes\, which can be used to create 1-cell spheres, rods, typical 2-cell, 4-cell or 8-cell stage embryos. In the DPP package, we further provide more detailed guidelines to create other shapes and constrains with SE. For the method description below we will use the example file “4 cells - confined in sphere - raw.fe”.

3.1.1.1 Creating shapes with Surface Evolver

1. Open Matlab. Navigate to the DPP Package folder in Matlab, and right-click on it and click “add to path/ Select folders and subfolders”. Navigate to the folder “DPP package\DPP program\” where three scripts named “launcher” are placed; you will work from here.
2. Open the script “launcher_1_generate_shapes.m” with Matlab. Set the path where you will store the shapes, simulation parameters and results. For instance, if you want to store your cells and their simulations in a folder “C:\Data\my cell” you set the variable: root_path = ‘C:\Data\my cell’.
3. Run the function that generates the folders: shape_paths = f1_generate_paths(root_path) by selecting the two above-mentioned lines and pressing F9 to evaluate/run. This will create the folder “C:\Data\my cell” and inside it a folder “C:\Data\my cell\shape_source”. In the latter you will keep all cell shape related files.

4. Copy the file “4 cells - confined in sphere - raw.fe” into the folder “C:\Data\my cell\shape_source” and double click the input file to open it in SE.
5. To see the surface, press “s” to enter the graphic mode (Fig. 1A). Then “q” to quit the graphic mode and return to SE command terminal. You may zoom in our out (press Z when graphic window is active) or rotate (press R when graphic window is active) the surface using the mouse.
6. To iteratively find minimal energy configuration (Fig. 1B), the user needs to perform subsequent refinements of the mesh and calculation of novel surface coordinates. Type “r” to refine the initial mesh (subdivide); this will increase the number of vertices. Then “g10” to make 10 iterations of surface minimization (g100 will make 100 etc.). Refine the mesh again with “r”. Refine and re-iterate calculations several times until you find that the shape is stable and does not change significantly with following iterations. As a start for division prediction, we recommend to refine the mesh to reach a number of typically 1000-2000 vertices.
7. Save the final file by typing “d” to save the data in one file in SE format. Use a different name than the original file, e.g. “4_cell_final”.
8. Type “export_lists()” in the SE terminal to extract the data in separate lists for vertices, edges, faces and bodies. Note that this function has been added to the example files to facilitate the following steps, but is not a default function of SE. In the package we provide this function which can be copied into any SE function. At the end of this process, you should end up with 2 SE files (.fe) one for the initial shape, and one for the final, and several .txt files for edges, faces, vertices and bodies.

3.1.1.2 Translating the SE surface into a Matlab matrix with labeled bodies.

1. In Matlab open the script “launcher_1_generate_shapes.m”. Find the function “f2_extract_bodies_from_SE(shape_paths, [X, Y])” and run it by selecting the line and pressing F9. The numbers in the brackets are the size in pixels of a Matlab matrix that will comprise the SE surface; for instance: [X, Y] = [200, 240] pixels. Any size in any proportion can be used, but large matrices may take much longer to simulate. Typically, we use matrices of the sizes ranging from 150x150 to 200x200 pixels; X and Y may be set unequal (Z is automatically recalculated from shape’s dimensions and given X and Y to ensure the best fit of the shape within the matrix).

2. Matlab will ask you to provide the name of the SE file from which you exported the lists (its name was used as a part of those lists and will be used to find them); in our case it is "C:\Data\my cell\shape_source\4 cells - confined in sphere - raw.fe". The program will read the raw coordinates in ".txt" files and save them as "C:\Data\my cell\shape_source\ shape_data.mat", after which it will translate them to matrix coordinates, label them according to the body lists and save those as "C:\Data\my cell\shape_source\ labeled_bodies.mat" in the same folder.
3. In Matlab open the script "launcher_1_generate_shapes.m". Find the function "f3_show_labeled_bodies(shape_paths)" and run it. It will find the file "labeled_bodies.mat" and show the labeled bodies (Fig. 1C). If everything went fine up to this point, the system is ready to run division predictions (Section 3.2).

3.1.2 Generating an input 3D shape from an imaging stack of a real embryo

In many instances, one may need to input the direct 3D shape from a cell or a group of cell in an embryo/tissue, and compare experimental division orientation with model predictions. Those shapes may be experimentally obtained from a Z-stack taken on a labelled embryo with confocal, light-sheet or 2-photon microscopy (Fig. 1D). Here we solely provide a method to convert a 3D segmented stack into an input file to run the division prediction program. Segmentation of 3D shapes may be achieved with ImageJ or Imaris, and will largely differ depending on the embryo, labels and microscopes. For our purpose, input segmented stacks must be black inside and white outside. The voxel should be cubic (it should span equally in all dimensions: $V_x = V_y = V_z$) and the final segmentation is recommended to span not less than 100x100x100 pixels (Fig. 1E).

In the folder "DPP package\DPP files\generating_shapes_from_experiment\" we provide an example stack of a segmented 4-cell Sea Urchin embryo called "4CellUrchin_final.tif" which we use to present this part of the method.

1. Open Matlab. Navigate to the DPP Package folder in Matlab, and right-click on it and click "add to path/ Select folders and subfolders". Navigate to the folder "DPP package\DPP program\" where three scripts named "launcher" are placed; you will work from here.
2. Open the script "launcher_1_generate_shapes.m" with Matlab. Set the path where you will store the shapes, generated cells and their simulations. For instance, if you want to store your cells and their simulations in a folder "C:\Data\my cell" you set the variable: root_path = 'C:\Data\my cell'. Run the function that generates the folders: shape_paths =

f1_generate_paths(root_path) by selecting the two above-mentioned lines and pressing F9 to evaluate/run. This will create the folder “C:\Data\my cell” and inside it a folder “C:\Data\my cell\shape_source”. Copy the file “4CellUrchin_final.tif” into this folder.

3. In Matlab open the script “launcher_1_generate_shapes.m”, find the function “f2_extract_bodies_from_IJ” and run it by selecting the line and pressing F9. It will offer to choose a segmentation stack; choose “4CellUrchin_final.tif”.
4. You will be taken to an interface allowing you to manually connect 2D regions in the stack to reconstitute the proper 3D shape of your embryo/tissue (Fig. 1F). For this, you will need to manually label 2D regions belonging to one cell from slice to slice (Fig. 1F, left). You can scroll through the whole stack using the mouse wheel. Then set a label number N, by pressing a numeric key on the keyboard and label a 2D region with this N number by clicking on each z-slice as it appears. The program takes the user to the next slice automatically after the click. After all slices of one cell have been connected, you can visualize the result in 3D by pressing <v> (Fig. 1F, right). After all cells have been reconstructed, it is recommended to clean the stack from lone pixels by pressing <c> and visualize in 3D again; the final result should look similar to Fig. 1F. Export the resulting bodies by pressing <s>. This will create a Matlab file “C:\Data\my cell\shape_source\labeled_bodies.mat”. If everything went fine up to this point, the system is ready to run division predictions (Section 3.2).

3.2. Predicting division position and orientation in 3D

In this part we describe the methods to use the scripts which allow to make a prediction on the orientation and position of the division axis. This prediction will have as inputs 3D cell shapes defined following the preceding paragraphs, and parameters, which relate to the distribution of astral microtubule (MT) forces within cells, and to simulation procedures. As such, it is important to state that those model can solely be useful to study cell division in cell types and systems in which MTs are prime contributor of nuclei/spindle position, as in most vertebrate and invertebrate early embryos [15,14]. In our model, MT forces may depend on the spatial distribution of MTs around centrosomes, the length of MTs, or the presence of cortical domains. MT length distribution are defined by cell shape and/or the presence or MT excluding structure in cells, such as neighboring asters or yolk.

3.2.1 Setting a choice of parameters for the simulations

1. In Matlab open the script “launcher_2_generate_cell.m”, which contains a set of parameters for the cell simulation. If you run this program following immediately sections 3.1.1.1 or 3.1.1.2, the path to your cell is already set. Else, redo the step 3.1.1.1.2 or 3.1.1.2.
2. Define cell to simulate: “body_ind =”. If you have several cells (like in the 4 cell embryo mentioned above), this is the index of the particular cell that will be used as the shape input. In the part described in section 3.1.2.4, to generate shapes from real 3D images, this number is the one defined by the user during the 3D reconstruction.
3. Define microtubule pulling force parameters. As described in [16], the general formula used to define the force exerted by each MT in the asters is:

   F = L^Beta1 + pullfactor*L^Beta2 + pullfactor_exp*2(L/Unit_branch)

   The first term corresponds to pulling in the cytoplasm in proportion to MT length, with an exponent factor, “Beta1” reflecting putative non-linearities, which by default is set to 3 [6]. The second and third terms reflect pulling from a polar cortical domain (whose location and size are defined hereafter). “Beta2” reflects an exponent for cortical pulling, and “pullfactor” the strength of this effect. Beta2 is set by default to 2 to reflect a situation in which diluted motors are limiting pulling from the domain [18]. The last term reflects a situation in which MT number at the domain are limiting pulling, and possesses and exponential dependence term, defined through the parameter “Unit_branch” , to mimic the presence of branching in large embryonic asters (a situation likely irrelevant for somatic cells). This parameter has a default value of 6.75, as experimentally measured in Sea Urchins [6]. The strength of this term is defined by the parameter “pullfactor_exp”. If both parameters “pullfactor” and “pullfactor_exp” are set to 0, then the script will not propose you to position and size a cortical domain (see hereafter section 3.2.1.7).
4. Define exclusion regions for microtubules growth. Microtubule effective length can be corrected for the presence of intracellular structure such as yolk which affects MT growth in a dose-dependent manner. This implies defining the location of the yolk gradient, its orientation, and its sharpness, as defined in [16]. The parameter “gradyolk” defines the sharpness of the yolk gradient (if set to 0, MTs are uniform inside the cell shape). “erfoffset” is the offset of the sharpest gradient point in pixels. “direction_grad” is used for the direction of the yolk gradient, and is defined by polar and azimuthal angles, in degrees, counted in a standard convention: θ_y is counted from the z-axis; φ_y is counted counter clock wise from the x-axis (Fig. 2B). “MTstab” defines the sensitivity of MTs to yolk, and is set to 1 by

default. Importantly, the shape of the yolk gradient will appear in the graphic interface upon running the script, and can thus be adjusted back (Fig. 2D and section 3.2.1.8).

5. Aster parameters. Parameters related to MT asters, include initial position and orientation of asters from where to start a simulation, extension angle of asters, and distance between centrosomes. “anglelimit_deg” reflects the extension of asters in degrees (β in Fig. 2A). “theta”, is the starting polar angle in spherical coordinates; counted from the z-axis (θ in Fig. 2A). “phi” is the starting azimuthal angle in spherical coordinates; counted counter clock wise from the x-axis (φ in Fig. 2A) . “startingxyz” is the initial position of nuclei/spindle center in pixels, defined as a by 3 cartesian coordnates [x, y, z]. If empty, the initial position will be at the calculated “centroid” of the cell shape. “Nucsize” is the distance between centrosomes in pixels (spindle or nuclear size). “Nucexcl” is a parameter used to calculate an exclusion volume ensuring that the nucleus/spindle stays inside the cell during the simulation. Nucexcl cannot be < 1; typical values are 1.2-1.4.
6. After all the parameters have been set, run the whole script by pressing the “Run” button in the Matlab Editor panel. This will create the parameter structures and feed them to the class “DPP_cell” needed for the simulation.
7. If you have set polarity parameters different than 0 (see 3.2.1.3), you will be offered to position and size a polarity domain. For this you will be brought directly to a graphic interface (Fig. 2C). The domain is defined as the intersection of the 3D cell shape with a cutting sphere. Use the interface to define values for the coordinates (x, y, z) of the sphere center and radius (R), and press “Cut” to visualize the domain. Once set, press the button “Save” to set the domain for the simulation. If you press “Quit” this will set the pulling pre-factors to 0 and run a simulation purely based of cytoplasmic MT forces and cell shape.
8. If everything was set correctly, the cell with polar domain, yolk gradient and initial orientation/position of nuclei/spindle will be shown as in Fig. 2D. Corresponding parameters will be automatically saved in a new folder “cell_n”. Each time you run this script the cell folder index “n” increases automatically, e.g. “C:\Data\my cell\cell_1”, “C:\Data\my cell\cell_2” and so on.

3.2.2. Starting a simulation

The principle of the simulation is based on a random walk around the 3D position and orientation of the nucleus/spindle by 3 center coordinates (x, y, z) and 2 angles (θ and φ). At each iteration one of those variable is changed, the force and torques of all MTs on the nucleus/spindle are calculated and compared to the previous step in order to converge towards an equilibrium of minimal forces and torques [16]. The simulation thus comes with parameters to optimize time, precision or accuracy, which need to be defined by the user.

1. Following the preceding section, open the script “launcher_3_generate_simulation.m”.
2. Set visualization parameters.

   With “show_log = 1” the program outputs to the Matlab command line information on what parameter has been varied and how forces are calculated. If not required, set to 0.

   With “show_visually = N” (higher than 0) the program will display the nucleus position and orientation each N iterations. If set to 0, there will be no visualization, which may improve simulation speed.
3. Auto-save flags. Set to 1 if needed and 0 if not required.

   “auto_save_nucleus_run = 1”: the program will save the run of aster centers (trajectory during simulation) as a PNG picture.

   “auto_save_torque_map = 1”: the program will save a torque map as a PNG (see below).
   “auto_save_state_history = 1”: the program will save the history of varying parameters during the simulation as a PNG.

   “auto_save_split_image = 1”: the program will save an image of a divided cell (with two daughter cells) as a PNG.
4. Random walk parameters.

   “angle_btw_mt_deg” defines the angle (in degrees) between microtubules (used for both MT polar and azimuthal angle), and somehow reflects the precision of the simulation. If set to 10, for instance, the forces will be calculated along MTs in the aster spanning each 10 degrees in both polar and azimuthal angles. The larger the angle between MTs, the faster the calculations are, but the less precise also. It is recommended to use large steps (10-20) to quickly and roughly find equilibrium position and then use lower angles (1-3) to refine this position and find proper orientation (see below).

   “jump_chance =” Probability of jumping to a new configuration, even if it has been estimated to be less favorable (forces and torques are higher) than the previous state. If set to 10, then the chance to make a jump is 10%.

   “Inertia =” is the probability to vary the same parameter in the same manner if its previous variation led to a more favorable state (100: always use previous variation; 0: randomly pick a new variation).

"duration_equilibrium =", number of iterations required for a new state to be tested for equilibrium. If x, y, z do not change for this number of iterations, a position is considered equilibrated.
"duration_full =" sets the maximum number of iterations (use typically > 5000).

5. The torque map is important as it informs on the stability of the current division orientation as compared to others. For instance in shapes which are roughly isotropic, this map would be flat with no sharp torque minima [16]. The torque map is a 2D matrix of absolute values of torques at each orientation angle (Fig 3D), calculated as:

$$T(\varphi_i, \theta_j) = \sqrt{T_{\varphi_i}^2 + T_{\theta_j}^2}$$

   Torque map parameters are:
   "torque_map_angle_step =" which sets a precision step for torque map calculation. If set to 10, torque map calculation will sample division orientations each 10 degrees for both θ and φ.
   "auto_calc_torque_map = 1": the program will calculate a torque map automatically after the simulation (it may take a long time depending on the precision of the map). If not required, set to 0.
6. Once parameters are set, start the simulation by selecting all the lines from "current_simulation" down to "current_simulation.run" and pressing F9.
7. The simulation will end either if it ran out of iterations (from the maximum number allowed), or if it found an equilibrium which did not move during a number of iteration set by the parameter inertia. The final state of the simulation will be saved automatically in the cell folder, i.e. "C:\Data\my cell\cell_n\" with the prefix corresponding to the index of current instance: "run_1", "run_2" and so on. Windows showing the results of the simulation will open, depending on the auto-save flags (see Fig. 3 for examples of simulation output).
8. If you want to refine the simulation, with more MTs to improve precision, or with a more detailed torque map etc. use the section "Refine division orientation", and insert new numbers, and run by selecting lines from "current_simulation.reset" down to "current_simulation.show_division_plane".
9. Finally, one can use old cell shape and parameters to re-run simulation. To that aim, open the folder with the cell of interest, e.g. "C:\Data\my Cell\cell_1" and drag and drop the file

"cell.mat" into Matlab workspace to load it. In the script "launcher_2_generate_cell.m" run the line "current_cell.show", visualize the initial state of the cell visually. You can also inspect all the properties of the cell by typing "current_cell" in the command line and pressing enter. Then restart from section 3.2.2.1 to run a new simulation.

Acknowledgments: We thank Anaëlle Pierre and Benjamin Blanchard for their initial contribution to the model. This research was supported by the CNRS and grants from the "Mairie de Paris emergence" program, the FRM "amorçage" grant AJE20130426890 and the European Research Council (CoG Forcaster N° 647073).

References

1. Gilbert S (2010) Developmental Biology. 9th edition., vol 9th Ed. Developmental Biology Sinauer Associates, Sunderland (MA)
2. Pelegri F, Dekens MP, Schulte-Merker S, Maischein HM, Weiler C, Nusslein-Volhard C (2004) Identification of recessive maternal-effect mutations in the zebrafish using a gynogenesis-based method. Dev Dyn 231 (2):324-335. doi:10.1002/dvdy.20145
3. Olivier N, Luengo-Oroz MA, Duloquin L, Faure E, Savy T, Veilleux I, Solinas X, Debarre D, Bourgine P, Santos A, Peyrieras N, Beaurepaire E (2010) Cell lineage reconstruction of early zebrafish embryos using label-free nonlinear microscopy. Science 329 (5994):967-971. doi:329/5994/967 [pii]

10.1126/science.1189428
4. Hertwig O (1884) Das Problem der Befruchtung une der Isotropie des Eies, eine Theory der Vererbung. Jenaische Zeitschrist
5. Hertwig O (1893) Ueber den Werth der ersten Furchungszellen fuer die Organbildung des Embryo. Experimentelle Studien am Frosch- und Tritonei. Arch mikr Anat xlii 662–807
6. Minc N, Burgess D, Chang F (2011) Influence of cell geometry on division-plane positioning. Cell 144 (3):414-426. doi:S0092-8674(11)00017-1 [pii]

10.1016/j.cell.2011.01.016
7. Wuhr M, Tan ES, Parker SK, Detrich HW, 3rd, Mitchison TJ (2010) A model for cleavage plane determination in early amphibian and fish embryos. Curr Biol 20 (22):2040-2045. doi:10.1016/j.cub.2010.10.024

S0960-9822(10)01288-1 [pii]
8. Neff AW, Wakahara M, Jurand A, Malacinski GM (1984) Experimental analyses of cytoplasmic rearrangements which follow fertilization and accompany symmetrization of inverted Xenopus eggs. J Embryol Exp Morphol 80:197-224
9. Yokota H, Neff AW, Malacinski GM (1992) Altering the position of the first horizontal cleavage furrow of the amphibian (Xenopus) egg reduces embryonic survival. Int J Dev Biol 36 (4):527-535
10. Sardet C, Paix A, Prodon F, Dru P, Chenevert J (2007) From oocyte to 16-cell stage: cytoplasmic and cortical reorganizations that pattern the ascidian embryo. Dev Dyn 236 (7):1716-1731. doi:10.1002/dvdy.21136

11. Dan K (1979) Studies on unequal cleavage in sea urchins I. Migration of the nuclei to the vegetal pole. Development, Growth and Differentiation 21 (6):527-535

12. Leonard JD, Ettensohn CA (2007) Analysis of dishevelled localization and function in the early sea urchin embryo. Dev Biol 306 (1):50-65. doi:S0012-1606(07)00172-8 [pii]

10.1016/j.ydbio.2007.02.041

13. Gonczy P (2008) Mechanisms of asymmetric cell division: flies and worms pave the way. Nat Rev Mol Cell Biol 9 (5):355-366. doi:10.1038/nrm2388

nrm2388 [pii]

14. Mitchison T, Wuhr M, Nguyen P, Ishihara K, Groen A, Field CM (2012) Growth, interaction, and positioning of microtubule asters in extremely large vertebrate embryo cells. Cytoskeleton (Hoboken) 69 (10):738-750. doi:10.1002/cm.21050

15. Hasley A, Chavez S, Danilchik M, Wuhr M, Pelegri F (2017) Vertebrate Embryonic Cleavage Pattern Determination. Adv Exp Med Biol 953:117-171. doi:10.1007/978-3-319-46095-6_4

16. Pierre A, Sallé J, Wühr M, Minc N (2016) Generic Theoretic Models to Predict Division Patterns of Cleaving Embryos. Developmental Cell 39:1-16

17. Brakke KA (1992) The Surface Evolver. Experimental Mathematics 1 (2):141-165

18. Grill SW, Hyman AA (2005) Spindle positioning by cortical pulling forces. Dev Cell 8 (4):461-465. doi:S1534-5807(05)00104-8 [pii]

10.1016/j.devcel.2005.03.014

Figures and Figure Captions

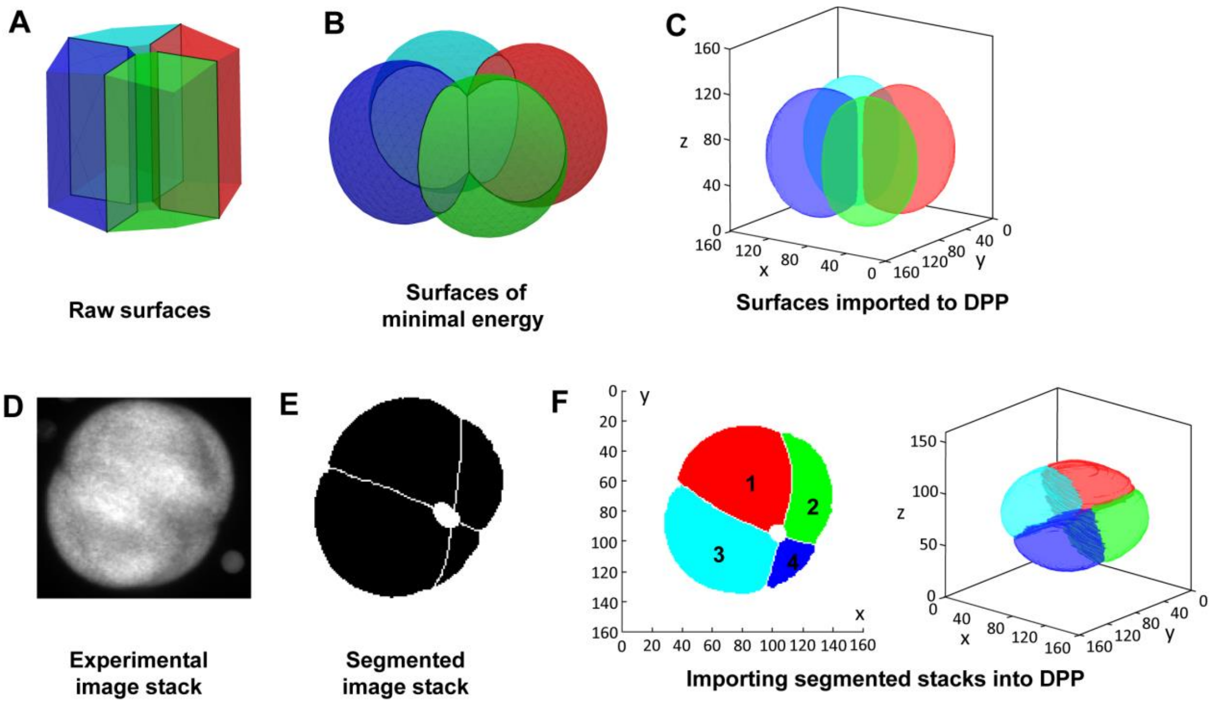
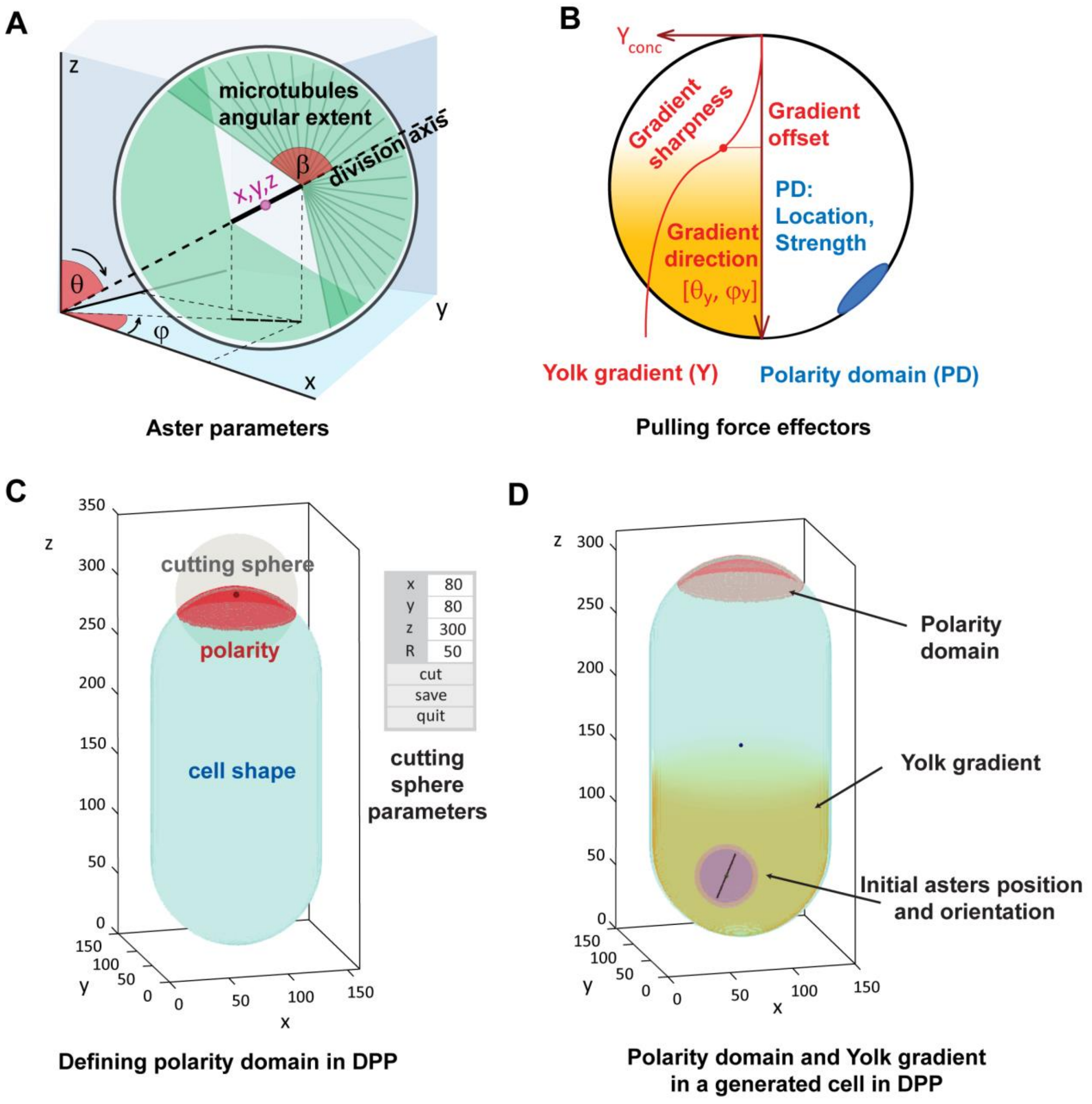


Figure 1. Creating 3D Cell Shape inputs. (A) Raw surfaces for a 4-Cell embryo in Surface evolver (B) Equilibrated shape with surfaces of minimal energy obtained in SE. (C) Equilibrated 3D shape imported to DPP with matlab. (D): Experimental image projected stack of a 4-cell sea urchin embryo; (E) Segmented image stack. (F): Interface to import 3D segmented stack in Matlab for DPP, through manual connection of 2D segments in 3D.



x	80
y	80
z	300
R	50

Figure 2. Input Parameters for predicting division position and orientation. (A): Microtubule aster parameters. Polar and azimuthal angles (θ and φ) defining the division axis angle; and position of the center o (x,y,z); angular extent of microtubules (β, counted from the division axis); (B): Parameters affecting microtubule pulling forces. MT growth may be tuned by the presence of yolk gradients (Y), with the concentration gradient defined by a direction (θ_y, φ_y), an offset, and a sharpness. A polarity domain (PD) with a given size, location and MT pulling strength may be also added as an input for simulations. (C) Matlab Interface to create a polarity domain: a cutting sphere of defined center and radius intersects the 3D cell shape to define the domain. (D) Visualization of the cell with its major simulation parameters used as initial inputs to predict division position and orientation.

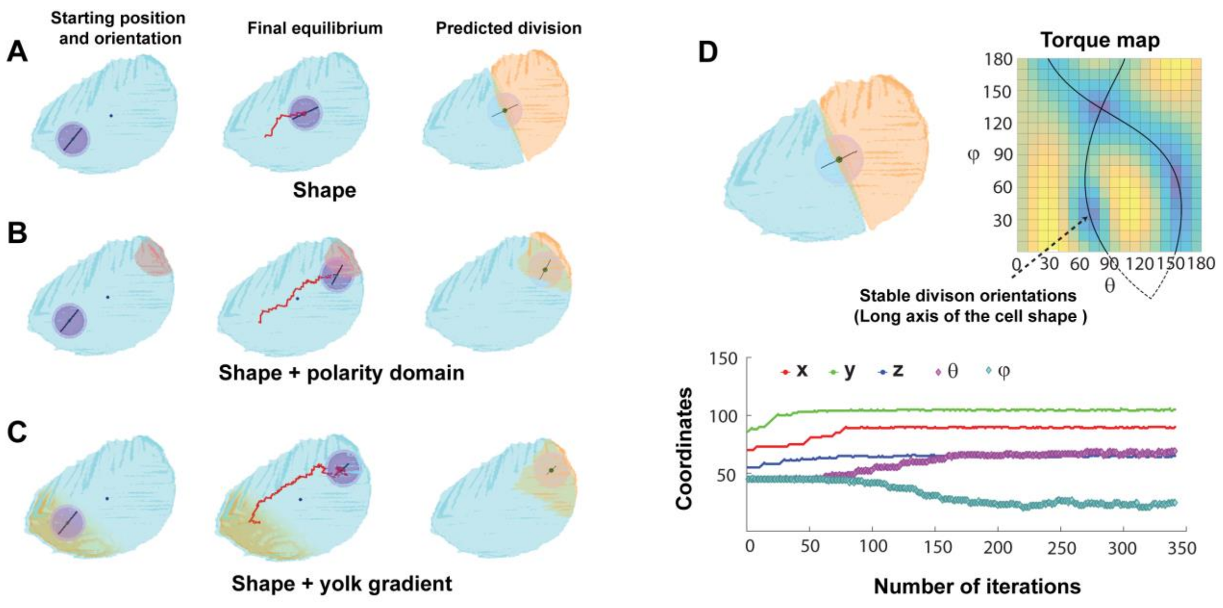


Figure 3. Example of DPP simulation results (A) (Left) Cell shape corresponding to one blastomere of the 4-cell embryo from figure 1D, is used as the sole input to predict division position and orientation. (Middle) After simulation, asters are position at the cell center and oriented along the long axis of the 3D shape. (Right) The predicted divided cell is then generated. (B) Same as is (A) but with the presence of a polarity domain pulling on asters. Note the strong asymmetry in cell division. (C) Same as in (A) but with a yolk gradient which creates a shift in the final position of asters away from yolk layers. (D) Simulation as in (A) with corresponding torque map, computing the torque exerted on asters as a function of all possible 3D angles. Torque minima correspond to stable orientations, highlighted by the black lines. (Bottom) Evolution of angular and position coordinates during the random walk, as a function of iterative step number of the walk.