Illustrating How Hydraulic Machinery Works

submitted by

Garoe Dorta-Perez

Research Project Preparation CM50175A

Unit Leader: Prof Yong-Liang Yang

University of Bath

Department of Computer Sciences

EngD in Digital Entertainment

January 2015

Summary

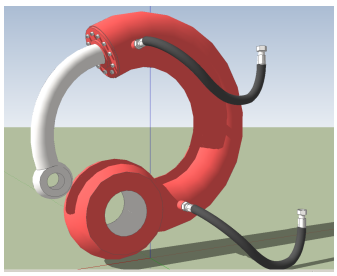
In this research proposal we present a method to semi-automatically depict *How things work* illustrations of hydraulic machinery.

Chapter 1

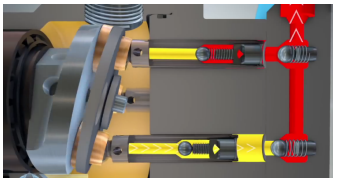
Introduction

How things work visualizations have been used as an efficient method to illustrate how a wide range of systems work. This technique usually involves displaying where each part is in relation to the system, showing how force is transmitted from one piece to the next or animating motion. In order to perform this task a range of visual transformations are used. Such as viewing the system from different angles, zoom degrees, transparency levels, as well as displaying only a subset of the parts. Generating material of this sort typically involves manual methods, usually in the form of an expert drawing each illustration by hand or composing a fixed animation using specific software.

(a) Cross section of a typical hydraulic cylinder.



(b) A 3D model of a hydraulic piston.



(c) Frame of manually generated animation of a hydraulic pump.

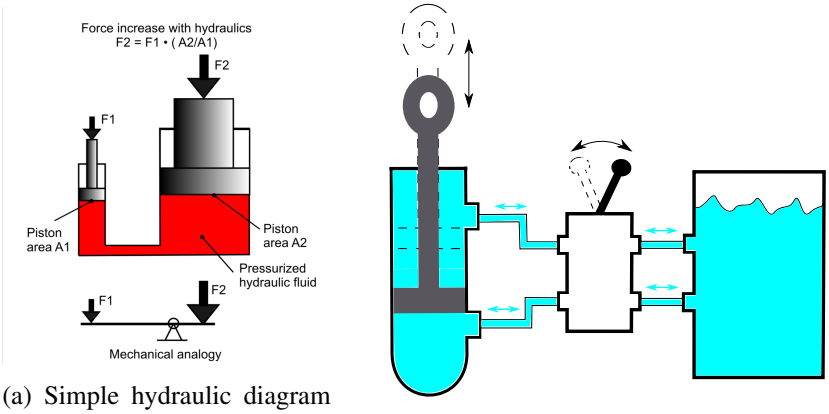
Figure 1-1: Overview of the intented workflow

Hydraulic machinery is commonly used in our everyday lives. Such as lifting cars with jacks, rams on excavators or gerotors to control fuel intake, as shown in Figure 1-1a. Their popularity is based on their faculty to transmit a force or torque multiplication independently

of the distance between input and output. Moreover, they operate silently, can wield force accurately and have the ability to apply torque multiplication easily. A typical hydraulic equipment has a contained liquid fluid that becomes pressurised when a force is applied to it, as shown in Figures 1-2a and 1-2b. Then that force is transmitted to the other end of the fluid. In order to grasp how the whole system works, it is essential to understand how the pressure is directed and how it interacts with other parts in the machinery. Therefore, to illustrate the general process, the spatial configuration of each part in the system must be unveiled. As well as the chain of motions that takes place within the gears and the hydraulic fluids.

In order to generate *How things work* visualizations, we first need a 3D model of the physical machinery that we will be working with, as shown in Figure 1-1b. There are some common visualisation and illustration techniques used for hydraulic machinery, as shown in Figure 1-1c. **Motion arrows** can point out how the solid parts move and they also indicate fluid flow movement. **Frame sequences** display key frames in complex motions and they can highlight temporal interactions between the parts. **Animations** are an useful tool for highly dynamic systems, for example when an excessive number of frame sequences are needed in a particularly complicated motion.

Generating visualisations and illustrations for hydraulic machinery is challenging for designers. They must understand in detail what forces are generated when parts interact with each other and what kind of flow movements are entailed. Furthermore, it is impossible to change the viewpoint to explore the object from different angle when 2D illustrations are used. Moreover, when animations for hydraulic visualizations are generated, they are infrequently updated, since manual animation is a costly and time consuming task.



(a) Simple hydraulic diagram with a mechanical analogy for easier understanding.

(b) Simplified hydraulic circuit for a hydraulic cylinder.

Figure 1-2: Sample diagrams for hydraulic mechanisms

There has been some work done on automatically generating illustrations and visualizations on mechanical assemblies. Nevertheless, it has been restricted to gear to gear interaction only. I.e. solid parts interacting with other solid parts, as gear movement can be inferred using symmetry information. Whereas work on fluid simulation and visualization has not been applied to hydraulic equipment. Simulations is this field are usually designed to, either generate complex visualizations for engineering purposes, or to produce visually plausible but not physically accurate results for animation or games.

This research proposal aims to introduce a method for generating *How things work* illustrations for hydraulic machinery. Simulations would be used to generate explanatory depictions of hydraulic machinery. This illustrations would help users understanding how this kind of equipment works.

In summary, the main contributions would be:

- An application for creating how things works illustration for 3D hydraulic machinery models.
- A method for detecting motion and interaction of fluid in the model parts.
- Algorithms to automatically generate illustrations with motion arrows and key frame sequences.

Chapter 2

Previous Work

2.1 Part analysis

Libraries of 3D mesh models are quite common [Tri14], [Gra14], [Aut14], in addition they contain vast quantities of materials. Usually the objects are also represented using points clouds, which can be obtained easily from the meshes. However, clouds and mesh models lack middle and high level information such as: symmetry or parallelism information. Therefore we must extract this data from either representation. Since solving for general shapes is quite challenging, a number of constrained approaches have been proposed. Moreover, part analysis goes beyond part segmentation, including part semantics and part relations analysis. For an overview on segmentation techniques, we refer the readers to the following surveys [VMC97], [APP*07] and [Sha08]. Whereas, for symmetry detection we recommend [MPWC13] survey.

2.1.1 Region growing

Region growing algorithms start with a seed element for the sub-mesh and then perform a local greedy element addition to the current cluster. [MDKK06] proposed a mesh segmentation technique based on curvature estimation with sharp edge recognition. The author's method performs well on mechanical objects and it is also robust to noise. However, oversegmentation occurs in models with soft edges. This model was extended by [LW08] with the use of Markov Random Fields. This probabilistic approach provides a optimal global solution using local interactions. Anyhow, the algorithm is not particularly efficient, taking several seconds to give a result.

2.1.2 Iterative clustering

Iterative clustering is based on the k-means clustering algorithm. Representatives for each cluster are chosen and then using some metric the elements are assigned to the best cluster and the representatives are recalculated. [LZHM06] presented a cluster based segmentation algorithm. The method performs better with hierarchical models composed of regions at multiple scales. Therefore, it gives the same results regardless of model scale or the coarseness of the mesh. A quadratic surface fitting algorithm was presented by [YWLY12]. Which combines several distance metrics to perform the quadratic minimization. Quadratic surfaces provide more flexibility than simpler representations, however the algorithm does not perform as well when the segments can not be represented by a quadratic surface.

2.1.3 Random walk

With the random walks approach, n faces are chosen as seeds, with n being the number of desired segments. Then for each unlabelled face, the probabilities that a random walker will reach each labelled face are calculated. Finally, each face is assigned to the highest probability label. [Gra06] proposed the first random walks implementation constrained into the imaged domain. Thereafter, [LHMR09] extended it to for 3D mesh models, the authors' technique also included an automatic seeding mode, where seed quantity and placement is based on a spring-like energy function. Seed placement can greatly affect the final segmentation. Nevertheless, the authors tackle the issue with a post processing stage were the clusters are merged if oversegmentation occurred and the edges are smoothed. Given its nature, random techniques give different results on each execution, the authors addressed the issue with their spring-like energy function, however their method is not completely invariant to seed placement.

2.1.4 Data driven

[GF09] presented an extension to the previous iterative clustering techniques with special emphasis on edge and face consistency. This started the data driven approach to segmentation. The algorithm is based on a graph construction which encodes triangle neighbour information, the method then performs a clustering on the graph space. The main advantage is consistent segmentation of similar models. For example, on different chair models, segments always represent legs, backs and armrests. However, there are several parameters that need to be input by the user. Moreover, as the technique is based on consistency across a set of models, if the alignment between the different models fails, the resulting segmentation quality will decrease.

Researchers have presented extensions to the original [GF09] approach, such as: setting a objective function as a Conditional Random Field model [KHS10].

2.1.5 Global energy function

[BLVD11] proposed a method based on a boundary edge function that would be learned by an AdaBoost classifier. Nevertheless, the algorithm needs a large database of manually segmented meshes, and it also has problems generalizing the function for complex test data. [dCdLL14] presented a novel segmentation technique based on calculating a global discrepancy function, based on the Lambert illumination model. For each triangle in the mesh a illumination value from several lights is calculated. Then they directly use that quantity to segment the model in several parts. However, their segmentation results are still not comparable with other state of the art methods.

2.1.6 Hybrid

[WY11] approach for mesh segmentation is based on a combination of curvature estimation, Gauss mapping and B-spline surface fitting. The authors' methods performs relatively well, however there are six numerical parameters to adjust and the presented results show oversegmentation with certain models. [YZW14] proposed a model to improve the normals calculation by adding joint information, and hence enhancing curvature detection in [WY11] method.

2.1.7 Slippage analysis

However all the previous methods cannot be directly applied to segment mechanical parts. [GG04] proposed a method for segmenting mechanical objects based on local slippage. This is quite useful as it gives for each part its degrees of freedom, e.g. sphere (3 rotations), cylinder (1 rotation and 1 translation), plane (1 rotation and 2 translations), etc. [XWY09] applied slippage analysis to perform joint-aware deformations in man-made objects. [YLT*14] improved [GG04] method, obtaining an algorithm more robust to noise and that would output extra primitive information (e.g. normal of a plane, center of a sphere, etc).

2.1.8 Symmetry detection

In man-made objects, and specially with assembly models, object symmetry can be used to detect important features in a part. With symmetric segments possible axis of rotation or translation can be computed. Additionally, non symmetric components normally imply movement constrains to the other parts.

[MGP06] presented an approximate symmetry detection method based on a voting scheme. The technique detects reflections, rotational symmetries and scaling. Mitra et al. later extended their previous work to automatic object symmetrization [MGP07]. However, parameters for symmetry detection and shape deformation still have to be manually input. An alternative approach to detect only rigid symmetries was proposed by [BBW*09]. Their method is based on feature lines, whose main advantage is better performance than previous methods. Nevertheless, their technique is constrained to only rigid transformations. [XZJ*12] presented a partial intrinsic symmetry detection system. Through a multi-scale voting scheme, intrinsic symmetries are extracted on multiple scales. Anyhow, erroneous symmetry fusion between different parts may occur when the sample points are on the same scale cluster.

2.1.9 Part characterization

Variational shape approximation has been studied for characterization purposes of engineered objects [CSAD04]. Volume based segmentation was explored by [AFS06], with a primitive fitting segmentation method. Nevertheless, the authors' method is not able to identify negative volumes, and it is not able to distinguish between the inner and the outer parts of the model. Building on previous work, [AMSF08] proposed an extension for region selection using polyhedra. [GSMCO09] demonstrated that man-made objects can be deformed intuitively with a set of 1D curves extracted from the models. An abstraction method, to extract characteristic curve networks was proposed by [MZL*09]. Which iteratively fits envelopes to the surface and performs a deforming simplification to smooth out details. Other techniques has been explored as well, such as planar based ones [MSM11]. However, their approach is limited to a learning phase with a manually marked training dataset. [YK12] proposed a controllable abstraction approach combining RANSAC and volume-centric primitive fitting to produce a range of simplified models. The algorithm is based on deforming the fitted primitives with a polynomial surface. An alternative to [AFS06] segmentation approach is also presented, addressing the previously outlined deficiencies.

2.2 Fluid simulation

The current paradigm in fluid simulation consist of solving the Navier-Stokes equations of fluid dynamics, shown below.

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

$$\mathbf{u}_t = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla \cdot (v\nabla\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}) \tag{2.2}$$

Equation 2.1 enforces mass conservation and ensures incompressibility. Equation 2.2 is derived from Newton's Second Law and encodes momentum conservation. Where $\mathbf{u}_t$ is the time derivative vector field of the fluid velocity, p is the scalar pressure field, ρ is the density of the fluid, v is the kinematic viscosity and $\mathbf{f}$ represents the body force per unit mass, usually gravity.

Along the years several methods have been proposed, although all of them are based on the aforementioned Navier-Stokes equations.

- **Fourier Transform** techniques uses several superimposed sinusoidal waves to model the fluid behaviour.
- **Grid** based methods track the fluid features at fixed points in space.
- **Smooth Particle Hydrodynamics** track a large number of particles in the fluid.
- **Hybrid** algorithms are combinations of the preceding methods.

Regardless of the chosen method, the fluid state has to be updated for each new frame. However, it is common for the algorithms to have a upper bound in the time step, so if the update is computed after the limit, a reasonable output is not guaranteed. Therefore, to avoid having to calculate extra in between updates that would not be rendered, techniques that allow for larger time steps are preferred.

Another important concept in fluid simulation is whether the simulation implements interplay within the fluid and rigid bodies that come in contact with it (solid-fluid coupling). And how flexible is this interaction, one way *solid-fluid* coupling (e.g. a rock drops on a small pool of water, so the water moves but the rock is not affected by the water reaction), one way

fluid-solid coupling (e.g. a small buoy floating in the ocean has little effect on the surrounding water), and *two way solid-fluid* interaction (e.g. a flexible object drops in a pool of fluid and both affect the behaviour of each other).

For more information on real time fluid simulations we refer the readers to [VLM12] and [IOS*14] surveys.

2.2.1 Fourier Transform

When simulating fluid with periodic boundary conditions, procedural simulation can be applied with low computational cost. Usually this method is used with large masses of water (e.g. ocean simulation), as they are an approximation to periodic boundary conditions fluids. The waves can be modelled as a superimposition of sinusoidal waves, which can be efficiently decomposed using Fast Fourier Transform methods [MWM87]. Waves produced in this fashion are visually plausible but physical accuracy is not enforced inherently by the model.

Each wave is characterized with a wave number k and a wave vector $\mathbf{k}$. [Tes01] presented the basic technique in this area. The authors modelled the ocean surface as a summation of complex sinusoids with different wave vectors $\mathbf{k}$. Visually pleasing results were achieved generating random Fourier amplitudes.

The previous method was further improved by [CGG03], with the addition of solids (e.g. ships, a coast line, etc). The model was extended through more complex wave shape and motion equations, with differentiated equations for open sea and near the shore situations. [CC06] further enhanced the method with the inclusion of adaptive surface tessellation. Furthermore, the method was implemented on GPUs, and it also included shallow water effects and spray dynamics. However, Fourier transform methods can still only be applied to simulate oceans or other huge masses of fluid.

2.2.2 Grid

Grid methods were among the first techniques to solve fluid simulation and so the firsts ones implement intuitive solutions. Grid based techniques solve Navier-Stokes Equations 2.1 and 2.2, in a fixed position in space (i.e. grid). As fluid flows, the equations are solved in each position, giving a value for speed and pressure at each time step. This approach is advantageous as many numerical methods can be applied easily on grids and they are more easily adaptable to GPUs implementations than other techniques.

One of the firsts papers in this area applied forward Euler integration to solve the Navier-Stokes equations in a grid [FM96]. However, their implementation of solid-fluid interaction was restricted to one way solid-fluid coupling, the solids did not affect the fluid behaviour. [Sta99] extended this method in order to overcome stability issues. The author's implemented a stable solver which allowed for larger time steps through an operator splitting scheme and more importantly a semi-Lagrangian method to calculate the advection term.

[CMT04] proposed a solid-fluid coupling algorithm for grids models using distributed Lagrange multipliers. Nevertheless, the author's method uses a non adaptive grid and multiphase fluids and bubbles are not covered by their implementation.

2.2.3 Smooth Particle Hydrodynamics

Smooth Particle Hydrodynamics(SPH) has been established as one the major breakthroughs for fluid simulations in computer graphics, positioning itself as the most popular method nowadays. Compared to grid based methods that need high resolution grids to produce foam and splashes, SPH can achieve the same with less computational load. Two extra advantages are that mass conservation is automatically satisfied, if the amount and mass of the particles is kept constant; and since the particles move with the fluid, the advection term does not need to be calculated explicitly.

SPH were first introduced by [Des96], where each particle encodes its position, velocity, mass and the forces that act on it. A globally adaptive time step is proposed as well, so the time step automatically adapts to the flow dynamics. What is more, to achieve more efficiency a local adaptive time integration is used, individual particles will use larger time steps if stability restrictions allow for it.

[MST*04] presented a method for computing solid fluid coupling where boundary particles are created on the surface of the solid and then particle coupling is calculated. [AIA12] further improved the previous method with the inclusion of irregular particle distributions as well as friction and dragging. [SZMTW14] solved stability issues in the previous SPH solid-fluid coupling techniques using a correction scheme. Furthermore the authors' algorithm was implemented on GPUs, thus achieving better performance than previous methods. However, the method is still not completely physically accurate and erroneous penetrations can arise in certain situations.

2.2.4 Hybrid

Hybrid methods are combinations of the aforementioned approaches. As with other hybrid techniques, the objective is to capitalize on the advantages of each, while trying to discard the weaknesses.

A common hybrid approach is the particle-in-cell (PIC) method, [Har62]. The author solves the advection using SPH and the incompressibility with an Euler grid. In more detail, the particles are averaged onto a grid where all the terms in the Navier-Stokes Equations 2.1 and 2.2 are calculated, except advection. Then, the particles are translated using the calculated grid velocity. This method has been used in a variety of situations, for example [ZB05] used it for sand simulation and [HG09] applied it for fire generation. A further improvement is an implicit approach, namely the fluid implicit particle (FLIP) [JH86]. The fundamental difference it that particles are not updated directly with the grid data. An advection term calculation that is always stable is the main strength of the FLIP method. However, the particles can sometimes clamp or have voids. FLIP was extended by [RWT11] to be able to solve the equations in a coarse grid. What is more, the authors' work also allowed for larger time steps, however a GPU implementation is not provided.

2.3 Flow visualization

Extensive work has been done in this area as visualizing fluid movement has a broad range of applications. However, this is a challenging task as it has to effectively display complex and copious amounts of data. Seeing that fluid simulation is generally solved by means of highly divided grids or with large number of particles, as explained in Section 2.2.

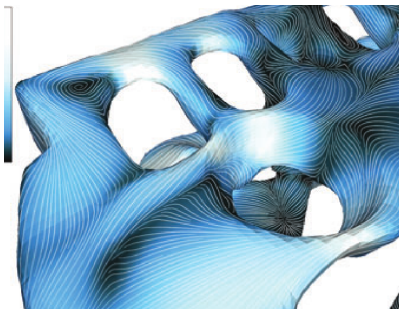


Figure 2-1: Streamlines on a 3D surface [SLCZ09].

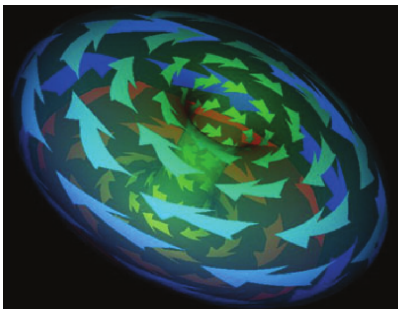


Figure 2-2: Arrows placed on streamlines paths in a 3D surface [Löf98].

Streamlines are convenient tools to describe and visualize flow, as shown in figure 2-1. A

streamline is defined as a curve that is everywhere tangent to flow field, i.e. it is parallel to the local velocity vector. Therefore, they provide an intuitive mechanism to show the fluid travel direction. Furthermore, they have properties such as: streamlines will not cross each other (flow will not go across them), or a particle in the fluid starting on one streamline will not leave it. Once the streamlines has been calculated, instead of displaying them as such, they can be replaced by arrows arranged using some criteria. For instance, on the path of each streamline or after clustering some streamlines together, as shown in Figure 2-2.

In the following sections we focus only on steady flow visualization. For more information on the subject we refer the readers to [MLP*10] survey.

2.3.1 2D flow

On the 2D image domain, an image-guided algorithm for visualizing 2D flow in images was proposed by [TB96]. The approach used inefficient seeding and streamline calculations. The first problem was addressed by [JL97], with a controllable seeding algorithm using two distance parameters. While the second was solved by [MAD05], the authors used a farthest seeding point strategy with a 2D Delauney triangulation. [LHS08] presented a novel method by only generating the fewest possible number of streamlines, while also preserving the most important flow features. Exploiting local coherence, similarity between streamlines is measured and used as criteria to disregard repetitive patterns.

2.3.2 Streamlines on surfaces

Seeding techniques for curves on 3D surfaces were explored by [vW92], who developed a particle based visualization method. Particles are seeded from geometric objects and continuous points sources will create streamlines. [MHHI98] proposed a technique to generate evenly spaced streamlines, extending [TB96] work. The algorithm maps 3D surface vectors in an extended 2D image domain and maps back the streamlines to 3D. [SLCZ09] presented a method to generate streamlines only for visible parts of the surface, thus providing a significant gain in efficiency. Moreover, the technique is able to handle general surfaces, adaptive resolution and supports exploration through user interaction.

2.3.3 Streamline rendering and placement

Rendering too many streamlines can result in clutter and too few can lead to omit important characteristics. There has been plenty research into optimizing streamline production and placement. However for our hydraulic machinery the issue is less critical than in other areas.

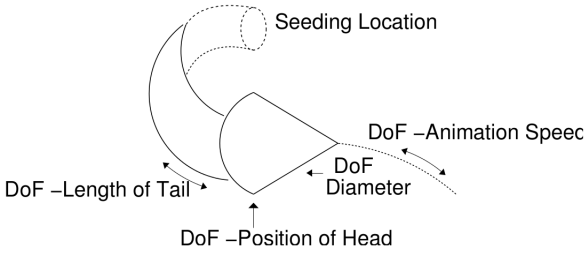


Figure 2-3: Streamcomet depiction with its degrees of freedom outlined [LH05].

[LH05] presented the streamcomet and a fast animating technique. A streamcomet, as shown in Figure 2-3, is an extension of a streamline, they move along the path of a stream line, with adjustable head position, length and translucency of tail, and variable animation speed. While the fast animating technique applies a stipple pattern to the streamline path, so the pattern is shifted at rendering time to add animation. [RPP*09] proposed a dual seeding technique for streamline placement. Dual streamlines that are orthogonal to the fluid flow are also calculated. However they are not rendered but rather used to better decide which and how the primal streamlines are going to be drawn. Considering the occlusion issues that normally arise in this area, [MCHM10] tackle the streamline rendering from a view dependant perspective. The authors precompute a random pool of streamlines. Then, occluded areas are pruned and empty areas are reseeded for the current view via occupancy buffers. What is more, a GPU implementation of their method is provided. Clustering is a common technique for reducing the amount of rendered streamlines. To address slow euclidean distance tests in clustering, [MJL*13] presented a signature based metric. The streamline attributes for the signature are curvature, torsion and tortuosity. The authors presented two variations, a fast simpler method and a hierarchical one that sacrifices speed for superior results. However, both provide a performance increase over euclidean distance calculations.

2.3.4 Surface based integral objects

Streamlines can be extended dimensionally in order to represent 3D surfaces instead of 3D lines, an example of a streamsurface is shown in Figure 2-4. Following this path leads to increased complexity in the methods, due to the new dimension needed to represent a surface.

Nevertheless, surfaces can provide better visualization results. For example, the use of shading gives an improved depth insight.

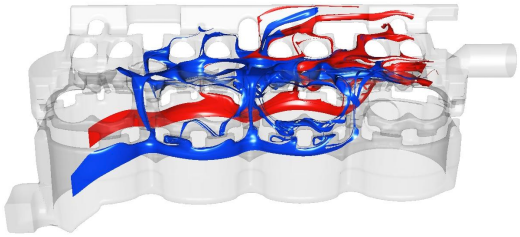


Figure 2-4: Streamsurface illustrating fuel flow in an engine [LGD*05].

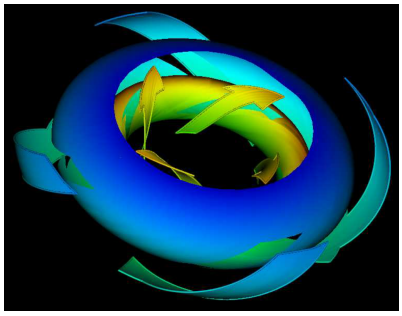


Figure 2-5: Streamarrows in an occluded flow situation [LMG97].

Streamsurfaces were introduced by [Hul92]. The author's method seeds streamlines from a curve and those streamlines are advanced through the vector field. New streamlines are seeded or exiting ones are terminated if the points reach a predetermined maximum or minimum distance threshold. The streamlines points are used to create the streamsurface mesh. To aid in the interpretation of flow, [LMG97] extended streamsurface model with the addition of streamarrows, as shown in Figure 2-2. The author's technique involves adding arrows to show flow direction, and removing the arrow space on the surface to help with occlusion. A hybrid method with streamlines and streamtubes was proposed by [ZDL03]. Depicting linear anisotropy regions with streamtube and planar anisotropy ones with streamsurfaces. The colors used for rendering encode additional anisotropy information. More recently, [BW10] presented extensions in streamsurface rendering to address occlusion issues, as well as a GPU implementation of their technique. Contour lines, halftoning, silhouetts, contour arrows, cuts and slabs are applied to a raw streamsurface in order to be able to depict occluded areas.

2.4 Explanatory illustration

Explanatory illustration has to adequately transmit motion on a still image, consequently transforming from the temporal space to the image domain. This is usually found in comics books and instructions sets.

2.4.1 Image and key frames output

Even though comics books are regularly considered inferior forms for presenting information, each scene contains abundant data and they can be understood by children without even reading the letters. [McC93] cited by [Cut02] reviewed visualization and abstraction techniques used in comics books. Such as, how to depict the motion of single objects and to illustrate noises and speeches bound to time. [Cut02] surveyed traditional techniques for depicting motion. Including, sequences of key frames, blur, squeeze and stretch, unbalanced postures and stroboscopic effects. The author introduced criteria to judge the effectiveness of a particular representation. For example, evocativeness, clarity of object, direction of motion and precision of motion.

[ND05] proposed a technique to depict motion in 3D animations. Scene and behaviour descriptions from specialized scene graphs were analysed in order to create the motion cues. In the author's implementation the user has to select a depiction target and assign labels to it, using the graph information motion lines and simplified sketch images are generated. [JR05] presented a similar approach for volumetric data. Speedlines and flow ribbons are combined to generate illustrations encoding 3D rigid movement and flow dynamics. The flow ribbons use comic inspired line abstraction technique to handle occlusion. Researchers have have look into automatic illustration generation for mechanical assemblies [MYY*10]. The author's work is based only on geometry information, extracted from 3D CAD models. Part analysis is performed using slippage analysis, see Section 2.1.7, while the motions are extracted from an interaction graph that is defined by edge contact between the components. The system is able to generate, motion arrows depicting mechanical assembly interaction, as well as generating key frames of periodic motions.

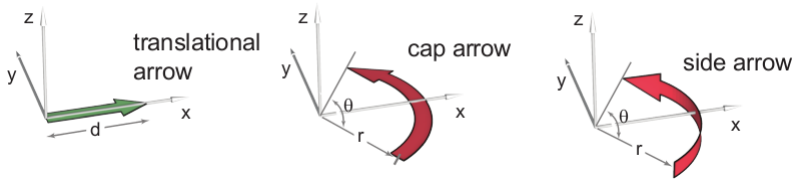


Figure 2-6: Example of arrows used to convey movement [MYY*10].

2.4.2 Video output

Video based illustration is applied to enhance the motion information while maintaining the original video format. Therefore, motions are exaggerated or at the very least some hints to

reinforce movement perception are added.

[CRH05] presented a method to embed cartoon-style motion cues in video. Their algorithm has two stages: a motion tracking step, where a feature manually marked will be tracked; and a computer graphics stages, where a cue will be generated and added to the video. The system can add motion lines as well as squash and stretch objects. However, the second case is restricted to videos where the background can be reconstructed from previous frames. [ZIH*11] presented a sketching system for fluids dynamics. The paper uses hybrid multi layered grid solver, discussed in Section 2.2.2, to solve 2.5D fluid flow. Flow paths and other objects can be added or changed in real time through a sketching interface. A hydraulic graph is created and is modified behind the scenes every time the visual model is edited. While the flow is shown using streamlines, discussed in Section 2.3. [Low14] showed that even though animations have become a generalized tool for visualizing dynamic systems, special care have to be taken as users can fail to extract the necessary information due to the nature of the animation.

Chapter 3

Methodology

3.1 Overview

Given a 3D CAD model of some hydraulic machinery we want to generate how things work visualizations. Namely, adding arrows depicting the fluid movement.

The topic can be subdivided into:

1. **Part analysis:** Detecting parts segments and inferring part properties.
2. **Fluid simulation:** Simulate how the fluid behaves in the previously detected parts.
3. **Flow visualization:** Display the fluid simulation data in a intuitive format.
4. **Explanatory illustration:** Transform the flow visualization data in one or several still images.

To start with, input data is needed in the form on CAD 3D models. A 3D visualization program such as SketchUp [Tri14] can be used to create, visualize and edit the models. Once we have this input, we can begin the process to generate *How things work* illustrations from them.

Part analysis involves two steps: part segmentation and part analysis. Usually 3D models consist of meshes or point clouds were there is not a clear distinction between each piece. Consequently, any additional analysis and simulation would be overly complicated without further simplifications. Therefore, a segmentation stage is needed in order to divide the model

into its constituent components. Part analysis consists of detecting the piece type, how it moves and interacts with others. So the information saved for type would be axle, gear, reservoir, fluid conduct, etc. In this classification parts are also organized considering fluid interaction criteria, e.g. if the piece interplays or not with the hydraulic fluid. With respect to types of movement, it will be direction of movement, axis of rotation, axis of translation, etc.

Once the parts have been categorized and given an input force, we will have to simulate how the force is transmitted along the different elements. In the special case where a component is a container of a fluid or is in direct contact with one, that force will have to be introduced in a fluid simulation algorithm. The output of the simulation will then carry the information along to the next piece.

In order to visualize the fluid simulation data we will need to generate a visual cue that will intuitively indicate the fluid movement. Either generating arrows indicating the overall fluid movement, with an animation or showing illustrative key frames.

Lastly, *How things work* illustrations have to be generated from the flow visualization data, that is by itself a time variant data. Therefore, techniques to embed motion in a single picture are used.

3.2 Proposed Methodology

3.2.1 Part analysis

As discussed in Sections 1 and 2, the input models for the system might not be segmented into parts. In Section 2.1 we presented several options to segment meshes. If we are to treat the more general case we need to perform a model to part segmentation. Volumetric extraction segmentation is needed to perform this task. Depending of the quality of the mesh a simpler method like [AFS06] or a more complex one [YK12] will be used. Then, the parts itself have to be segmented to obtain more information, such as type of the part or rotation axis. Slippage analysis [YLT*14] is the only method that can be applied directly to mechanical machinery. Moreover, this method outputs the degrees of freedom for each part. However, symmetry analysis [MGP07] can give more detailed information about part rotation and translation axis, and to extract extra features such as edges or gear radius. 1D features curves are useful for characterizing these and other attributes of man-made parts [GSMCO09]. Therefore, a geometric analysis can be performed and we can work with models that do not have extra time information embedded. Which is the case for most of the models available in the model libraries in

Section 2.1. However, none of the methods will automatically detect which parts will interact with the hydraulic fluid. A first approach will be to manually mark them. Nevertheless, other paths can be explored, such as heuristics that analyse the segment shape. For example, the biggest cylindrical part will probably be the reservoir for the fluid, and the parts connected to it have high probability to have the hydraulic liquid flowing through then.

3.2.2 Fluid simulation

Once we have segmented the model and we know how each part can move and where can the hydraulic fluid flow, we need to calculate how the liquid interacts with the solid parts. The previous part analysis is specially useful in order to simplify the simulation stage. If we already know how the solid parts can move, the simulation will be simpler since each segments will have constrained degrees of freedom. Besides, extra information about the parts,such as maximum translation range, that otherwise would have to be input manually would be automatically available to the fluid simulation. Given the characteristics of our input, a physically accurate simulation with two way solid-fluid coupling is needed. Fourier methods are discarded because we we will not be modelling vast amounts of fluid. Then, we are left with either SPH, grid or hybrid simulations. In terms of implementation complexity, grid methods are generally simpler, followed by SPH and lastly by hybrid systems. If we look at quality of the simulation, hybrid and SPH tend to be better while grid methods lag behind. However, the great disadvantage of grid simulations is the need to compute the simulation in highly divided grids in order to be able to display foams and other effects. What is more, SPH have become quite popular in the last years as a general purpose fluid simulation tool. Nevertheless, for hydraulic machinery the fluid will be constrained to a small area. Moreover, extra realistic effects, such as foams or droplets, are irrelevant for this proposal. Therefore, the previous grid disadvantage is no longer applicable, making methods such as [CMT04], the more appropriate type of simulation for our purposes.

3.2.3 Flow visualization

The next step is to generate the flow visualization from the simulation data. Following the techniques discussed in Section 2.3, we will use streamlines to visualize the flow. *How things work* illustrations do not have complex flow visualizations, as the objective is to understand roughly how the system works, not to give a perfect scientific representation of fluid dynamics. Since our CAD models are 3D, 2D methods are ruled out. There is not clear advantage in applying surface based integral objects as in hydraulic equipment there are not complex fluid

flow situations. Streamcomets offer a more natural way to visualize the flow, and having a translucent body means that occlusion is less likely to happen. However, we consider streamcomets too complex for our purposes, therefore a simpler technique such as [WST09] would be used, with the optimizations presented by [MJL*13] should the generation be too slow. This flow visualization should also include an optimization step, where the arrow placement would be improved. Following the criteria in [MYY*10], steps such as increasing arrow size in zones with more flow, replacing arrows in occluded areas or adding extra arrows in bigger areas, will be necessary to improve illustration quality.

3.2.4 Solid parts visualization

Solid parts visualization will be based on [MYY*10] methods for mechanical assemblies. This stage involves placing arrows to convey part movement on the edge of each junction. Different arrow types will be used, such as the ones shown in Figure 2-6. Moreover, extra arrows will be added if the parts are too long and, if occlusion were to happen the arrow placement can be optimized. However, if this is not enough, the type of the arrow could be switched if it produced an increased in visibility, from cap to side or viceversa.

3.2.5 Key frames visualization

To create key frame visualizations a simple approach is to sample the animation created from the fluid simulation at key points [MYY*10]. In order to select those key points there are two cases: objects with cyclical movement and parts that do not. The first ones would be spinning components, for example cylinders in engines whose movement is a periodic loop. In this case, a displacement curve can be drawn and the sample points would be the extrema and middle frames in between. In machinery that does not present cyclical movement, such as the simple cylinder shown in Figure 1-1a, it is evident that the extrema have to be included. However, we are faced with a problem of either under or oversampling in the middle points, as the minimal number of key frames to illustrate the motion are unknown.

3.3 Timeline

In Figure 3-1 we present an estimated timeline for the implementation of the proposal. This estimation is highly dependant on the features to be implemented and the complexity of the

CAD input models. The input of one task is the output of the one preceding, so overlapping naturally occurs when we are finishing one task as moving towards the next. We inevitably have to fix unexpected errors or introduce new features in the former task. Moreover, the changes may backtrack several steps backs as shown in the isolated bars of Part Characterization or Fluid Simulation.

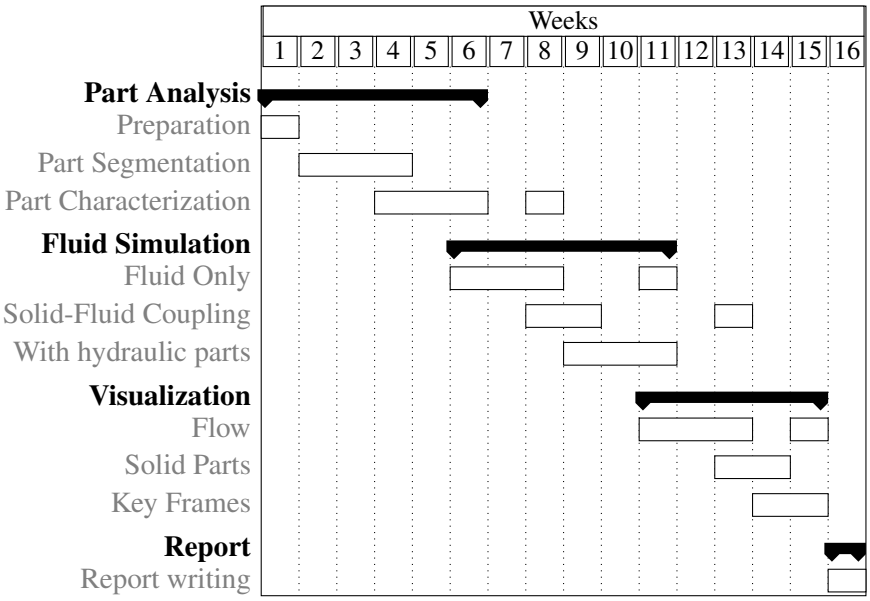


Figure 3-1: Timeline for a four months completion of the proposal.

3.4 Evaluation

The system will be tested by generating illustrations for a wide variety of models. Since we do not know of any previous work which has generated *How things work* illustrations for hydraulic machinery, comparison with other methods is limited to contrasting our work against manually generated illustrations. Being able to replicate a series of chosen hydraulic illustrations could be a goal of the project. Another path to evaluate the effectiveness of the generated data is to perform user studies. One example could be an evaluation on how well the test subjects understood the machinery internal workings with a series of questions, to estimate their knowledge on the equipment before and after seeing the illustrations. Another could measure which illustration the user finds as more instructing, the automatically generated ones or manually generated, as well as asking them why they think in such a manner. Moreover, user studies can be helpful in fine tuning parameters in the algorithm, therefore improving the overall qual-

ity of the results. Comparing frame sequences, animations and single illustrations is another potential evaluation study.

Chapter 4

Applications and Conclusions

We are proposing a tool whose main aim are educational purposes. Therefore, the most obvious applications would be to generate *How things work* visualizations for museums, schools and educational centres in general. Such institutions would benefit from having a illustration generator tool. Since by its semi-automatic nature, it will address the issues with depictions created manually by experts that were discussed in Chapter 1, such as a time consuming generation stage or lack of actualization when new machinery arrives.

We have presented a proposal for a semi-automatic *How things work* illustrations generator for hydraulic machinery. The system would work with 3D CAD models of hydraulic assemblies and it would output the illustrations, either single frame or key frames. The work flow of the algorithm includes a segmentation stage, were the 3D model is divided into parts with restricted degrees of freedom, a fluid simulation stage were the hydraulic interactions are calculated, and finally a visualization stage were techniques from flow visualization and illustrating motion in single images are used to depict the overall internal workings of the hydraulic machinery.

Bibliography

[AFS06] Attene M., Falcidieno B., Spagnuolo M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (Feb. 2006), 181–193.

[AIA12] Akinci N., Ihmsen M., Akinci G.: Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics 31*, 4 (2012), 62:1–62:8.

[AMSF08] Attene M., Mortara M., Spagnuolo M., Falcidieno B.: Hierarchical Convex Approximation of 3D Shapes for Fast Region Selection. *Computer Graphics Forum 27*, 5 (July 2008), 1323–1332.

[APP*07] Agathos A., Pratikakis I., Perantonis S., Sapidis N., Azariadis P.: 3D Mesh Segmentation Methodologies for CAD applications. *Computer-Aided Design and Applications 4*, 6 (Jan. 2007), 827–841.

[Aut14] Autodesk: Autodesk 123D, 2014. URL: http://www.123dapp.com/3d-model-library.

[BBW*09] Bokeloh M., Berner A., Wand M., Seidel H.-P., Schilling A.: Symmetry Detection Using Feature Lines. *Computer Graphics Forum 28*, 2 (Apr. 2009), 697–706.

[BLVD11] Benhabiles H., Lavoué G., Vandeborre J.-P., Daoudi M.: Learning Boundary Edges for 3D-Mesh Segmentation. *Computer Graphics Forum 30*, 8 (Dec. 2011), 2170–2182.

[BW10] Born S., Wiebel A.: Illustrative stream surfaces. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6 (2010), 1329–1338.

[CC06] Chiu Y.-f., Chang C.-f.: GPU-based Ocean Rendering. In *Multimedia and Expo, 2006 IEEE International Conference on* (July 2006), IEEE, pp. 2125–2128.

[CGG03] Cieutat J., Gonzato J., Guitton P.: A general ocean waves model for ship design. *Virtual Concept 4*, 7 (2003), 187–194.

[CMT04] Carlson M., Mucha P. J., Turk G.: Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 377–384.

[CRH05] Collomosse J., Rowntree D., Hall P.: Rendering cartoon-style motion cues in post-production video. *Graphical Models 67*, 6 (Nov. 2005), 549–564.

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. *ACM Transactions on Graphics 23*, 3 (2004), 905–914.

[Cut02] Cutting J. E.: Representing motion in a static image: constraints and parallels in art, science, and popular culture. *Perception 31*, 10 (2002), 1165–1194.

[dCdLL14] de Castro P. M. M. a., de Lima L. A. P., Lucena F. L. A.: Invariances of single curved manifolds applied to mesh segmentation. *Computers & Graphics 38*, 0 (Feb. 2014), 399–409.

[Des96] Desbrun, Mathieu and Gascuel M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation 96*. Springer, 1996, pp. 61–76.

[FM96] Foster N., Metaxas D.: Realistic Animation of Liquids. *Graphical Models and Image Processing 58*, 5 (Sept. 1996), 471–483.

[GF09] Golovinskiy A., Funkhouser T.: Consistent segmentation of 3D models. *Computers & Graphics 33*, 3 (June 2009), 262–269.

[GG04] Gelfand N., Guibas L. J.: Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing - SGP '04* (New York, NY, USA, 2004), ACM Press, pp. 214–223.

[Gra06] Grady L.: Random walks for image segmentation. *IEEE transactions on pattern analysis and machine intelligence 28*, 11 (Nov. 2006), 1768–1783.

[Gra14] GrabCAD: GrabCAD, 2014. URL: http://grabcad.com/library.

[GSMCO09] Gal R., Sorkine O., Mitra N. J., Cohen-Or D.: iWIRES: An Analyze-and-edit Approach to Shape Manipulation. *ACM Transactions on Graphics 28*, 3 (July 2009), 33:1–33:10.

[Har62] HARLOW F. H.: *The particle-in-cell method for numerical solution of problems in fluid dynamics*. Tech. rep., Los Alamos Scientific Lab., N. Mex., 1962.

[HG09] HORVATH C., GEIGER W.: Directable, high-resolution simulation of fire on the GPU. *ACM Transactions on Graphics 28*, 3 (July 2009), 41:1–41:8.

[Hul92] HULTQUIST J. P.: Constructing stream surfaces in steady 3D vector fields. In *Proceedings of the 3rd conference on Visualization'92* (1992), IEEE Computer Society Press, pp. 171–178.

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH fluids in computer graphics. In *Eurographics 2014-State of the Art Reports* (2014), no. 2, The Eurographics Association, pp. 21–42.

[JH86] J.U. BRACKBILL, H.M. RUPPEL: FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics 65*, 2 (1986), 314–343.

[JL97] JOBARD B., LEFER W.: Creating evenly-spaced streamlines of arbitrary density. In *Visualization in Scientific Computing 97*. Springer Vienna, 1997, pp. 43–55.

[JR05] JOSHI A., RHEINGANS P.: Illustration-inspired techniques for visualizing time-varying data. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 679–686.

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics 29*, 4 (July 2010), 102:1–102:12.

[LGD*05] LARAMEE R. S., GARTH C., DOLEISCH H., SCHNEIDER J., HAUSER H., HAGEN H.: Visual analysis and exploration of fluid flow in a cooling jacket. In *Visualization, 2005. VIS 05. IEEE* (2005), IEEE, pp. 623–630.

[LH05] LARAMEE R., HAUSER H.: Geometric flow visualization techniques for CFD simulation data. In *Proceedings of the 21st spring conference on Computer graphics* (2005), vol. 1, pp. 221–224.

[LHMR09] LAI Y.-K., HU S.-M., MARTIN R. R., ROSIN P. L.: Rapid and effective segmentation of 3D models using random walks. *Computer Aided Geometric Design 26*, 6 (Aug. 2009), 665–679.

[LHS08] LI L., HSIEH H.-H., SHEN H.-W.: Illustrative Streamline Placement and Visualization. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific* (Mar. 2008), IEEE, pp. 79–86.

[LMG97] Löffelmann H., Mroz L., Gröller E.: Hierarchical streamarrows for the visualization of dynamical systems. In *Visualization in Scientific Computing 97*. Springer, 1997, pp. 155–163.

[Löf98] Löffelmann H.: *Visualizing Local Properties and Characteristic Structures of Dynamical Systems*. PhD thesis, Technical University of Vienna, 1998.

[Low14] Lowe R. K.: Dynamic Visualizations: A Two-Edged Sword? In *Handbook of Human Centric Visualization*, Huang W., (Ed.). Springer New York, New York, NY, 2014, pp. 581–604.

[LW08] Lavoué G., Wolf C.: Markov Random Fields for Improving 3D Mesh Analysis and Segmentation. In *3DOR* (2008), pp. 25–32.

[LZHM06] Lai Y.-K., Zhou Q.-Y., Hu S.-M., Martin R. R.: Feature sensitive mesh segmentation. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling - SPM '06* (New York, NY, USA, 2006), ACM, pp. 17–25.

[MAD05] Mebarki A., Alliez P., Devillers O.: Farthest point seeding for efficient placement of streamlines. In *Visualization, 2005. VIS 05. IEEE* (2005), pp. 479–486.

[McC93] McCloud S.: *Understanding Comics-The invisible art*. Harper Perennial, New York, NY, USA, 1993.

[MCHM10] Marchesin S., Chen C.-K., Ho C., Ma K.-L.: View-dependent streamlines for 3D vector fields. *IEEE transactions on visualization and computer graphics 16*, 6 (2010), 1578–86.

[MDKK06] Mizoguchi T., Date H., Kanai S., Kishinami T.: Segmentation of Scanned Mesh into Analytic Surfaces Based on Robust Curvature Estimation and Region Growing. In *Geometric Modeling and Processing-GMP 2006* (2006), Springer, pp. 644–654.

[MGP06] Mitra N., Guibas L., Pauly M.: Partial and approximate symmetry detection for 3D geometry. *ACM Transactions on Graphics (TOG) 25*, 3 (2006), 560–568.

[MGP07] Mitra N. J., Guibas L. J., Pauly M.: Symmetrization. *ACM Transactions on Graphics (TOG) 26*, 3 (2007), Article 63.

[MHHI98] Mao X., Hatanaka Y., Higashida H., Imamiya A.: Image-guided streamline placement on curvilinear grid surfaces. In *Visualization '98. Proceedings* (1998), vol. 98, IEEE, pp. 135–142.

[MJL*13] McLoughlin T., Jones M. W., Laramee R. S., Malki R., Masters I., Hansen C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE transactions on visualization and computer graphics 19*, 8 (Aug. 2013), 1342–1353.

[MLP*10] McLoughlin T., Laramee R. S., Peikert R., Post F. H., Chen M.: Over Two Decades of Integration-Based, Geometric Flow Visualization. *Computer Graphics Forum 29*, 6 (Sept. 2010), 1807–1829.

[MPWC13] Mitra N., Pauly M., Wand M., Ceylan D.: Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum 32*, 6 (2013), 1–23.

[MSM11] McCrae J., Singh K., Mitra N.: Slices: a shape-proxy based on planar sections. *ACM Transactions on Graphics 30*, 6 (2011), 168:1–168:12.

[MST*04] Müller M., Schirm S., Teschner M., Heidelberger B., Gross M.: Interaction of fluids with deformable solids. *Computer Animation and Virtual Worlds 15*, 34 (July 2004), 159–171.

[MWM87] Mastin G. A., Watterberg P. A., Mareda J. F.: Fourier synthesis of ocean scenes. *Computer Graphics and Applications, IEEE 7*, 3 (1987), 16–23.

[MYY*10] Mitra N. J., Yang Y.-L., Yan D.-M., Li W., Agrawala M.: Illustrating How Mechanical Assemblies Work. *ACM Trans. Graph. 29*, 4 (July 2010), 58:1–58:12.

[MZL*09] Mehra R., Zhou Q., Long J., Sheffer A., Gooch A., Mitra N. J.: Abstraction of man-made shapes. *ACM Transactions on Graphics 28*, 5 (Dec. 2009), 137:1–137:10. URL: http://portal.acm.org/citation.cfm?doid=1618452.1618483, doi:10.1145/1618452.1618483.

[ND05] Nienhaus M., Dollner J.: Depicting dynamics using principles of visual art and narrations. *Computer Graphics and Applications, IEEE 25*, 3 (2005), 40–51.

[RPP*09] Rosanwo O., Petz C., Prohaska S., Hege H.-C., Hotz I.: Dual streamline seeding. In *2009 IEEE Pacific Visualization Symposium* (Apr. 2009), IEEE, pp. 9–16.

[RWT11] Raveendran K., Wojtan C., Turk G.: Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation* (2011), pp. 33–42.

[Sha08] Shamir A.: A Survey on Mesh Segmentation Techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556.

[SLCZ09] Spencer B., Laramee R. S., Chen G., Zhang E.: Evenly Spaced Streamlines for Surfaces: An Image-Based Approach. *Computer Graphics Forum 28*, 6 (Sept. 2009), 1618–1631.

[Sta99] Stam J.: Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), pp. 121–128.

[SZMTW14] Shao X., Zhou Z., Magnenat-Thalmann N., Wu W.: Stable and Fast Fluid-Solid Coupling for Incompressible SPH. *Computer Graphics Forum* (Oct. 2014).

[TB96] Turk G., Banks D.: Image-guided streamline placement. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 453–460.

[Tes01] Tessendorf J.: Simulating ocean water. *Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH 1* (2001), Course Notes 47.

[Tri14] Trimble Navigation Limited: 3D Warehouse, 2014. URL: https://3dwarehouse.sketchup.com/.

[VLM12] Vines M., Lee W.-S., Mavriplis C.: Computer animation challenges for computational fluid dynamics. *International Journal of Computational Fluid Dynamics 26*, 6-8 (July 2012), 407–434.

[VMC97] Várady T., Martin R. R., Cox J.: Reverse engineering of geometric modelsan introduction. *Computer-Aided Design 29*, 4 (Apr. 1997), 255–268.

[vW92] van Wijk J.: Rendering surface-particles. In *Visualization, 1992. Visualization '92, Proceedings., IEEE Conference on* (1992), pp. 54–61.

[WST09] Wicke M., Stanton M., Treuille A.: Modular bases for fluid dynamics. *ACM Transactions on Graphics 28*, 3 (July 2009), 39:1–39:8.

[WY11] Wang J., Yu Z.: Surface feature based mesh segmentation. *Computers & Graphics 35*, 3 (June 2011), 661–667.

[XWY09] Xu W., Wang J., Yin K.: Joint-aware manipulation of deformable models. *ACM Trans. Graph. 28*, 3 (2009), 35:1–35:9.

[XZJ*12] Xu K., Zhang H., Jiang W., Dyer R., Cheng Z., Liu L., Chen B.: Multi-scale partial intrinsic symmetry detection. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 181:1–181:11.

[YK12] YUMER M. E., KARA L. B.: Co-abstraction of shape collections. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 166:1–166:11.

[YLT*14] YI B., LIU Z., TAN J., CHENG F., DUAN G., LIU L.: Shape recognition of CAD models via iterative slippage analysis. *Computer-Aided Design 55*, 0 (Oct. 2014), 13–25.

[YWLY12] YAN D.-M., WANG W., LIU Y., YANG Z.: Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design 44*, 11 (Nov. 2012), 1072–1082.

[YZW14] YANG X., ZHENG J., WANG D.: A computational approach to joint line detection on triangular meshes. *Engineering with Computers 30*, 4 (Dec. 2014), 583–597.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972.

[ZDL03] ZHANG S., DEMIRALP C., LAIDLAW D. H.: Visualizing diffusion tensor MR images using streamtubes and streamsurfaces. *Visualization and Computer Graphics, IEEE Transactions on 9*, 4 (Oct. 2003), 454–462.

[ZIH*11] ZHU B., IWATA M., HARAGUCHI R., ASHIHARA T., UMETANI N., IGARASHI T., NAKAZAWA K.: Sketch-based Dynamic Illustration of Fluid Systems. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (2011), ACM, pp. 134:1–134:8.