Proposal: RoboVis

Alistair Wick (alistair.wk@gmail.com)

6th March 2017

Abstract

I propose a Visualization tool which may be used to explore the space of possible configurations for a 3-DOF robotic arm, with the aim of allowing newcomers to the design to find configurations which meet their requirements. The focus of the tool will be on the load the arm can carry, how that load is distributed spatially, and where the arm is able to reach. Crucially, the tool should provide some form of guided exploration, showing not just how a given configuration will perform, but how the performance will change as the configuration is altered in different ways.

Domain

The domain of the problem is small-scale robotics, and more generally mechanical design. One of the goals will be to make the tool accessible to those with limited domain experience. Ultimately, the tool would sit as the user-facing component of a system which permits the rapid design and creation of customized robot arms.

Users would be anyone with a potential use-case in mind for a small robotic arm: for example, hobbyists, or automation engineers. The only requirements on potential users would be a basic understanding of how the arm would move, which could be conveyed by a short animation or description, and some fundamental principals of mechanics: for example, the meaning of a "load", and a basic understanding of torque as a measure of rotary "strength". This is because the user would have to be able to translate their use-case into rough figures for what their customized arm should be capable of (i.e. "picking up cans" to "500g load at 20 to 35cm distance") as well as understanding the values that the vis tool both requires as input and displays as output.

Task

The main task is to find a configuration which matches the user's requirements: following the task abstraction guidelines in *Visualization Analysis and Design* (Munzner 2015), this is a parameter-space search/explore task, since neither the "location" (configuration) nor the *exact* target (arm capabilities) are known ahead of time. Notably, the hyperdimensional nature of the problem makes providing an overview of the exploration space difficult. What is the target of the search? Since the space is continuous, any viable user requirements for an arm will typically yield an infinite number of configurations which are acceptable; however, the target of the search is simply any single configuration which fulfills the requirements, not the entire set.

The tool might also find related uses, outside of this core task: for example, one might wish to explore the space to discover the limits of practical designs.

On the spectrum of "specific" to "general" vis tools, this is fairly specific: it aims to solve one problem well.

Dataset

I intend to work with a dynamically-generated dataset: I have Python code which can find the reachable points for a given configuration, which will form the basis of the dataset. A small amount of additional work will be needed to calculate the maximum force loading for each point, and I hope to be able to optimize the Python code to the point that it runs at near-realtime speeds; this would be beneficial for, but not essential to the development of the vis tool.

The configuration space – the space to be explored – is essentially a hyperdimensional field, where any point is itself a multi-value continuous 2D field, describing the reach and performance of the arm corresponding to the configuration values which index said point. Naturally these will have to be discretized: my working idea is that the configuration will start with an intial "seed", and the visualization will display possibilities for configurations at discrete steps (along selected dimensions) away from the current one. The output 2D field(s) can be discretized for display by sampling in a grid.

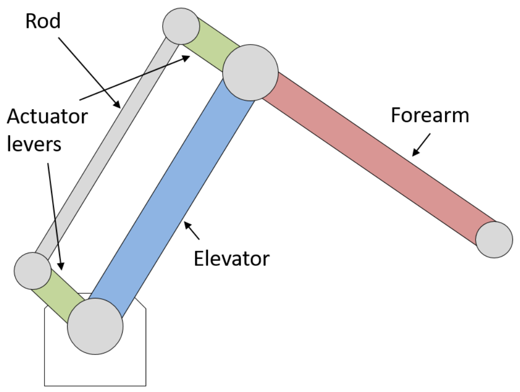

Figure 1: Basic layout of the arm

The configuration will consist of *at least*:

Attribute	Description
Elevator length	First section of the arm – see diagram
Forearm length	Second section
Actuator ratio	Ratio of lengths of the upper and lower actuator levers; affects leverage in the design
Rod ratio	Ratio of length of the actuator rod to the elevator; affects leverage
Elevator torque	Maximum torque available to drive the elevator; limits the maximum load, and should include any gearing of the drive mechanism
Actuator torque	Maximum torque available to drive the actuator, which indirectly drives the forearm; also limits the maximum load

Where every attribute is a continuous, quantitative, sequential value.

Additional attributes might include angular limits, for example between different sections of the arm, or on the drive system. Many more attributes are possible, and would not represent a significant challenge algorithmically; they would simply be additional constraints in the solver. The problem is that visualizing these extra attributes effectively will be difficult. I intend to start with the most important attributes – those listed above – and go from there.

Motivation

3D printing of parts for a robotic arm is a fairly common idea, but using a fixed design is a missed opportunity: in 3D printing, there is no time cost (on the fabrication side) to changing the design of a part with every print, so highly customizable objects become practical. These can be generated algorithmically, according to some end-user input or requirements, using software like OpenSCAD.

In the case of a robotic arm, a generic mechanical design could be modified by, for example, changing the lengths of the different parts of the arm, or altering the gear ratios in the drive system. This is conceptually simple, but the changes in the capabilities of the arm which derive from these configuration changes are not necessarily straightforward. For example, in my design, a parallel actuation mechanism can generate leverage which changes according to the respective lengths of the different parts, while at the same time expanding or constricting the area which the arm can physically reach.

With this difficulty in mind, *RoboVis* is intended to help come up with useful configurations for a generic 3-DOF design.

Personal Expertise

The robotic arm design I will be using is my own, one I developed in my final year at Bristol University; as such, I already have a fair amount of experience with the design and mechanics of the arm. I wrote the Python control code which keeps the arm within its operational limits (fig. 2), and it is this control code (or rather, the code which maps out the reachable area) which makes this project possible. My confidence that I will be able to extend and optimize the code (and thereby the dataset) comes from the fact that I am the one who wrote it, and so I am well-qualified to modify it.

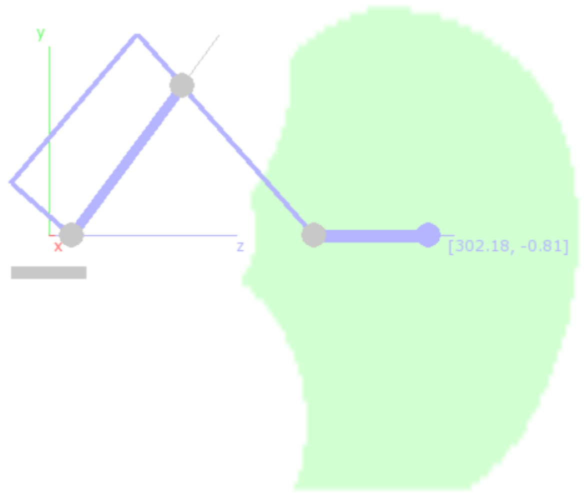

Figure 2: “Reach” visualization in the *PyIK* tool used to control the robot arm

I have some experience developing 2D graph-like desktop applications, including familiarity with typical graphics concepts like transformations and the scene graph. This makes me confident I will be able to implement the vis tool with a framework like Qt (through PyQt), which provides powerful 2D scene graph rendering utilities.

Proposed Solution

I present two options for interaction modes in the vis tool, geared towards different levels of latency in data retrieval. Both options would use largely the same graphical user interface (fig. 3), centred around a spatial display of the arm's reach for a given configuration, or set of configurations. I avoid small-multiple displays – as in (Keefe et al. 2009) – since I believe understanding the exact shape and scale of the reachable area is essential to the task, and it seems that the irregular and unpredictable nature of these shapes means they would not lend themselves to visual comparison between multiple, small side-by-side views.

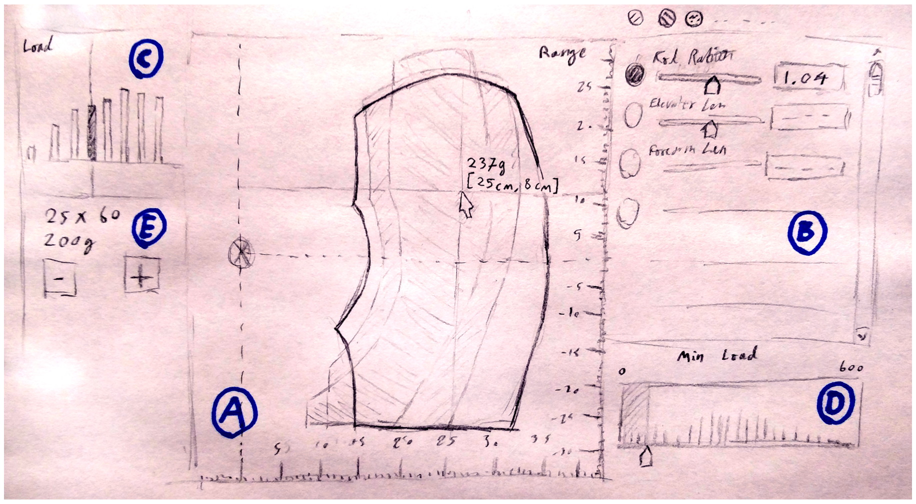

Figure 3: Sketched overview of the vis tool

Option 1: Realtime

In this scenario, I am able to optimize the solver to run at near-realtime speeds, producing at least a partial solution set for display in less than 0.1 second (following the *Visualization Analysis & Design* rules of thumb). This approach could tolerate some latency by using asynchronous calculation, showing the results as they become available, but asynchronicity is not a panacea; if latency prevents smooth interaction on my development machine, I will use the "option 2" approach for interaction.

Essentially, interaction would revolve around the iterative change of a "current" configuration, with the goal of getting closer to (and eventually meeting) the goal requirements. This is a take on "Informed trial and error" in (Sedlmair et al. 2014).

Range

The tool would center around a large display (fig. 3 A) showing the arm's spatial reach, as an outline, for the current configuration. When attributes are selected for inspection, additional colored outlines would be drawn in-place, with the color corresponding to the attribute selection. These outlines would show how the range would change with changes to the selected attributes, with a divergent saturation/luminosity scale showing the *direction* of the change – positive or negative (fig. 4).

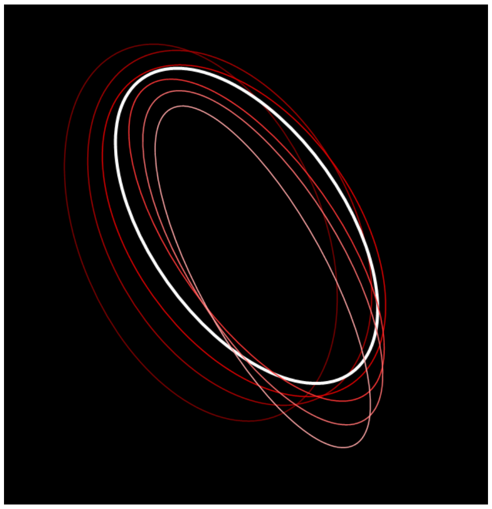

Figure 4: Simplified representation of a graded outline vis; the shapes will be more complicated than rings in the real tool

Changing the current configuration would immediately redraw the main display, creating the impression of smoothly "sliding" through configurations. Ruler markings at the edges of the display, and a coordinate readout at the cursor would allow the user to judge real-world scale.

Load Capacity

Each configuration's load capacity would vary across its range of operation, making it difficult to display the capabilities of more than one configuration at once. A toggle option would enable or disable a "heatmap" layer, showing the maximum load for the current configuration (displaying this layer would partially or fully obscure the additional dynamic outlines), and a small histogram widget, part of a scented slider, would display the distribution of the capacity for the configuration.

Realtime Scenario of Use

I will assume the user has some idea of what they need; examples might be:

- Lift 400g items from a low to high position, without much lateral movement
- Move 100g items within a (1m^3) volume
- Position a 600g timelapse camera in a (20cm^3) volume

When the user opens the tool (this is likely to take the form of a desktop application), they will be presented with a view corresponding to a "default" configuration for the arm. They then follow a rough loop to refine the configuration to their needs.

Evaluation: They examine the current configuration. The arm's reach is evaluated primarily through the main display, which shows this in an intuitive spatial fashion. Precise dimensional information is available through interaction, where the user hovers the cursor over the display to see the exact coordinates of the hovered point in real-world units.

The load is examined by switching on a "heatmap" display, or simply examining the load distribution widget (fig. 3 D). Again, precise information is available by hovering the cursor at a point of interest for a numerical readout of the maximum load.

Alteration: The user is then able to dynamically select one or more numerical attributes they are interested in exploring in the right-hand panel (fig. 3 B), by dropping a colored "puck" next to those

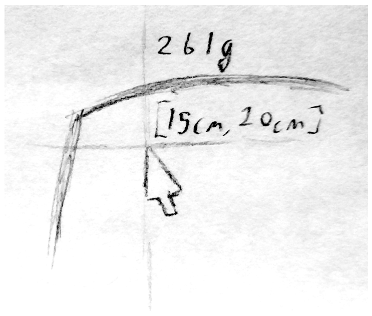

Figure 5: Coordinate and load capacity readout on hover

attributes. The main display then updates with additional outlines, mapping the ranges for configurations which have the chosen attribute shifted in either direction; the coloration of the new outlines matches the color of the chosen puck(s).

If the user is interested in adjusting for the max load, rather than the range, they can hover different points in the main display while looking at the second load widget (fig. 3 C), which then shows a comparison of the max load at the hovered point for the current and "adjacent" configurations.

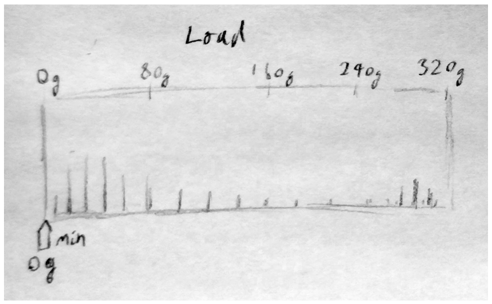

Figure 6: Scented load capacity slider adjusts the minimum allowable load capacity, and displays the distribution of capacity as a histogram for the current configuration.

If the user knows they have a "floor" on the load capacity, below which the arm is not useful, they can use a scented slider (fig. 6) to adjust the minimum allowable value, triggering a redraw of the displayed outlines, now limited both mechanically and by the minimum allowed load; that is, the outlines are shrunk to show only the areas within which the arm can both reach physically, and carry loads greater than the setpoint. The scented slider shows a histogram of the frequency of load capacity samples (corresponding to the area they cover) over the range of 0 to the maximum load capacity. This provides additional contextual information over a standard slider (Willett, Heer, and Agrawala 2007), and allows the user to quickly see how much of the overall range they will be removing by increasing the minimum allowed load.

To rapidly "scale" the design up or down, the user presses the scaling buttons (fig. 3 E), which respectively increase or decrease the values of all length and power parameters at once, resulting in a larger or smaller configuration which can carry approximately the same load as before.

After making an alteration, the user returns to the "examination" stage and reevaluates the new configuration.

Option 2: Batched

If I am unable to optimize the solver sufficiently for a realtime application, I will opt for a subtly different approach. The essentials of the view would remain the same, but with realtime "sliding" adjustment being impossible, a focus would be placed on iterating through "nearby" configurations. Calculations would need to be asynchronous, to avoid disrupting interactions with the tool.

The approach discussed here is a take on the "Local-to-Global" approach in (Sedlmair et al. (2014)).

Batched Scenario of Use

Examination: Same as above

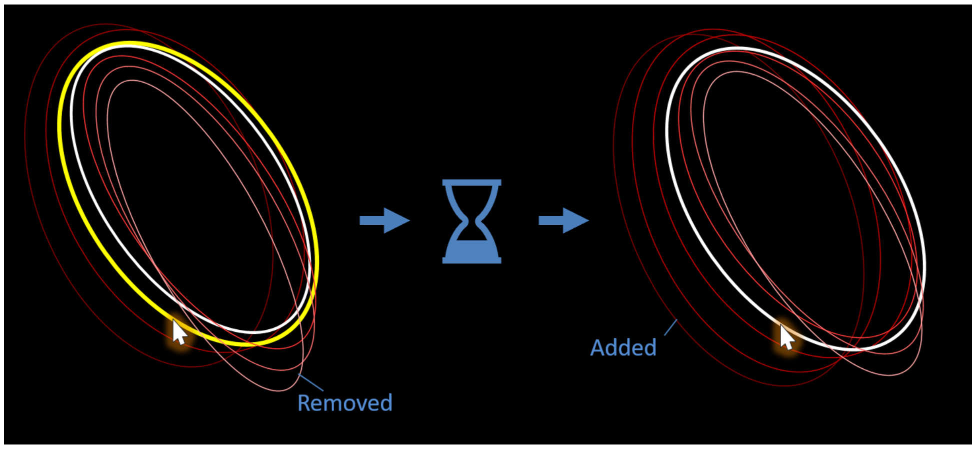

Figure 7: Hover/highlight followed by selection of new focus/center configuration

Selection/Alteration: Attributes are selected as above, and the user waits while the display updates; outlines are calculated and displayed with increasing offsets from the current attribute values. At any point, the user can highlight an outline by hovering over it (to make the outline clearly visible against the rest), and select it by clicking on it, at which point it becomes the "current" center configuration. "Distant" configurations are then removed, and new outlines are again generated outward from the selected one.

Rapid scaling would be performed in the same way as above.

Milestones and Schedule

Prep: 06-14th March

Initial work setting up a project framework and transferring the "dataset" code to this framework. The end of this stage will represent a strict cutoff for any extensions and optimizations to the dataset code, as well as the point to decide between the "realtime" or "batched" options for the vis design.

During this time, I will also be making the final choice of the development tools I will be using – while I will certainly use Python on the data side, there are options on the vis side. Choices include using a PyQt graphics view, a PyQt web canvas, or linking an in-browser D3 vis to a local Python server. Again, the decision will be locked in by the end of this period.

Implementation I: 15th-27th March

This will include implementation of basic visualization components, including the main display and its hover interaction, as well as the load capacity widgets.

Here I will also be making early choices about what works, and what doesn't, in order to guide further implementation work. I will discuss my findings, and any resulting decisions in the interim report. An important focus here will be the "outline" visualization – specifically, how many different attributes can be rendered at once while maintaining legibility.

Interim report/evaluation: 28th-31st March

Implementation II: 1st-19th April

In the final stretch of implementation, I will work on making the tool as full-featured and refined as possible, adding any of the interaction features that were missed in the first stretch. I will also begin my evaluation of the tool's performance and utility.

Presentation and writeup: 20th-28th April

References

Keefe, Daniel F, Marcus Ewart, William Ribarsky, and Remco Chang. 2009. "Interactive Coordinated Multiple-View Visualization of Biomechanical Motion Data" 15 (November). doi:10.1109/TVCG.2009.152.

Munzner, Tamara. 2015. *Visualization Analysis & Design*. CRC Press.

Sedlmair, M, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller. 2014. "Visual Parameter Space Analysis: A Conceptual Framework" 20 (December). doi:10.1109/TVCG.2014.2346321.

Willett, W, J. Heer, and M. Agrawala. 2007. "Scented Widgets: Improving Navigation Cues with Embedded Visualizations" 13 (November). doi:10.1109/TVCG.2007.70589.