**Fu et al.(animated construction)** demonstrates a method to estimate drawing order from static line drawings with clearly defined shape geometry. Mainly, it maps the key principles of drawing order such as i) coarse-to-fine hierarchical representation of an input line drawing ii) order a set of drawing strokes and finally iii) determine directions for each individual stroke. They address the coarse-to-fine method by clustering and approximation. a clustering step for reducing the number of curves, and an approximation step for reducing shape complexity of individual strokes. They achieve stroke ordering via energy function where it depends on length of the stroke, distance between the closest points (proximity), and angular difference between endpoint tangents (collinearity). A na¨ıve solution involving simple enumeration of all the possible drawing orders is expensive O(n!). So, they approximate the solution using a Hamiltonian graph minimization. Although the Hamiltonian path problem is NP-complete, given a carefully constructed graph, the problem search space is significantly smaller than that from a na¨ıve construction.

Empirically, they observed that the order of detail strokes is less important and thus they resort to a simpler strategy instead of the computationally expensive strategy as required for the significant lines. The detail lines in a group approximate collection by certain similarity (**as we do**) or by semantics. And, the stroke direction is solely determined by the acute angle between the line defined by the two end points of the stroke

However, the order of compilation of drawings might vary with people. Therefore they’re focusing on one that looks plausible to human viewers. They consider only line art images with cleanly defined lines or curves and exclude line drawings with shading or texture, complex geometric sketches where it’s become hard to distinguish from one another.

The computationally expensive step involves finding Hamiltonian paths on k-NN graphs, whose running time ranges from a few seconds to 2- 5 minutes, depending on the value of k, the number of significant lines, and the configuration of k-NN graphs. And, in complex ones (e.g., Figure 8 camera and boy) manual intervention may be needed. They assume the inputs to be relatively clean, free of hatching strokes.

**Tong et al.(Vector Flow and Grayscale)** implements image-to-pencil with a drawing process by drawing one stroke at a time. They divide the pencil sketching task into two steps. For the first step, they develop a parameter-controlled pencil stroke generation mechanism based on the pixel-scale statistical results of some real pencil drawings. For the second step, they develop a framework to guide strokes arranging on the canvas. To draw a straight line, they specify three parameters: central pixel gray value mean G, line width W, and line length L and they use we the edge tangent flow vector field to guide the direction of the strokes; the gray image (quantization result) to determine the location, length, and shade of the strokes; the edge map for detail enhancement.

However, the ETFs don’t have inherent sequence order. The representation of pencil lines is one form of sketching and doesn’t support a wide variety of sketchings/paintings.

**Huang et al** (**learning to paint**) adopted model-based Deep Deterministic Policy Gradient (DDPG) algorithm to train a neural agent to learn to do oil painting with process. Deep reinforcement learning (DRL) requires a massive amount of parameters when training, so the network’s input size is very limited. It can only handle 128 128 images, unable to generate fine-grained details, while our algorithm has no restriction on the size of the input image and could generate high-quality details.

**Liu et (Dynamic Sketching)** al investigates the problem of simulating the process of observational drawing, that is, how people draw lines when sketching a given 3D model. They present a multi-phase drawing framework and

the concept of sketching entropy, which provides a unified way to model stroke selection and ordering, both within and across phases.

They propose to simulate the drawing process in a sequence of phases. The drawing actions are ordered phase by phase, like posture phase, primitive phase, contour phase, and finally detail phase. Posture lines are not about accuracy or likeness but to capture the pose and action. They model these lines as a 3D curve-skeleton of the model, which can be extracted from the segmentation information. Primitive phase starts with drawing a profile hull, which roughly captures the space of one or more object parts. In the contour phase, the parts are integrated into a complete object. The perceptual saliency is fully revealed in detail. To reflect the hierarchical nature of the human drawing process, they propose to order the lines in the four phases at three levels: first phase by phase, then part by part, and finally stroke by stroke. They measure the information gain between previously drawn strokes before and target drawing as the ground truth and builds a graph as in Fu et al but adopts the greedy Prim’s minimum-spanning-tree algorithm

to extract the minimum cost edges. The effectiveness of the proposed approach relies on the quality of input segmentation. In addition, the drawing phases are designed based on the prior knowledge and cannot be automatically customized for different users.

A statistical analysis shows that our entropy-based ordering strategy leads to more plausible results than those driven by the conventional Gestalt rules used in previous work [Fu et al. 2011]. And it's not clear how to integrate common sketching behaviors, such as refinement and retracing, into their framework.

Jain et al (Multi-layer stencil) investigated a method to generate a set of layers for stencil creation. Though it’s not a drawing process, it presents the painting's evolution. Mainly, it uses multi-label semantic segmentation and the order of each layer is determined by energy function.

SketchRNN and Paint transformers are other relevant approaches.