An Automated Grading/Feedback System for 3-View Engineering Drawings using RANSAC

Youngwook Paul Kwon
UC Berkeley
Berkeley, CA 94720
young@berkeley.edu

Sara McMains
UC Berkeley
Berkeley, CA 94720
mcmains@berkeley.edu

ABSTRACT

We propose a novel automated grading system that can compare two multiview engineering drawings consisting of three views that may have allowable translations, scales, and offsets, and can recognize frequent error types as well as individual drawing errors. We show that translation, scale, and offset-invariant comparison can be conducted by estimating the affine transformation for each individual view within drawings. Our system directly aims to evaluate students' skills creating multiview engineering drawings. Since it is important for our students to be familiar with widely used software such as AutoCAD, our system does not require a separate interface or environment, but directly grades the saved DWG/DXF files from AutoCAD. We show the efficacy of the proposed algorithm by comparing its results with human grading. Beyond the advantages of convenience and accuracy, based on our data set of students' answers, we can analyze the common errors of the class as a whole using our system.

Author Keywords

Autograder, multiview engineering drawing, affine transformation estimation, RANSAC.

ACM Classification Keywords

I.4 Computing Methodologies: IMAGE PROCESSING AND COMPUTER VISION

INTRODUCTION

Multiview drawing is an international standard "graphical language" to represent 3D objects with 2D drawings. By following the rules of the graphical language, people can communicate the shape of three-dimensional objects without ambiguity. A multiview drawing consists of orthogonal projections to mutually perpendicular planes, typically the *front*, *top*, and *right* views. In the U.S., these are arranged on the page using so-called third angle projection, as if orthogonal projections onto the sides of a transparent glass box containing the object had been unfolded onto the page [1]. The three typical projections of a simple 3D object under third angle projection are shown in Figure 2. Sometimes additional projections are drawn for interpretation convenience. At the University of California at Berkeley, multiview drawing is taught in the lower division course "Basic Engineering Design Graphics," Engineering 28 (E28).

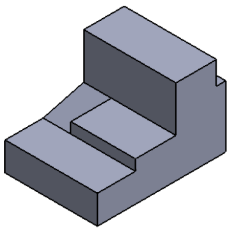


Figure 1. 3D geometry represented in multiview drawings in Figure 2-4.

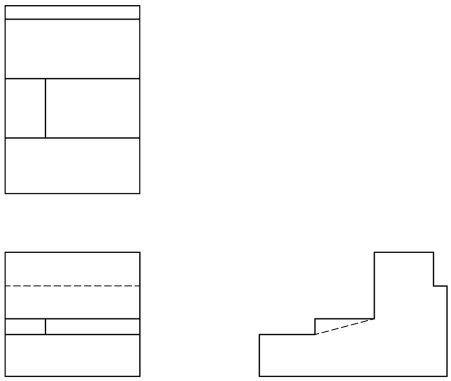
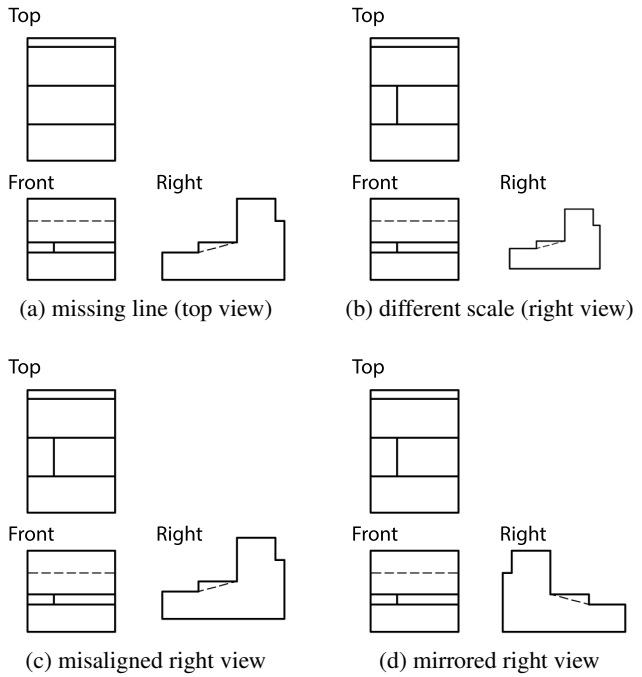


Figure 2. An example of a formal multiview drawing. Note that in multiview engineering drawings the views are not labeled; the placement and alignment communicates the relative viewpoints.

Due to the fundamental importance of engineering drawing for design and communication, E28 is a large class serving students majoring in fields including mechanical engineering, electrical engineering, computer science, industrial engineering, civil engineering, nuclear engineering, and architecture. Manually grading students' multiview drawing submissions and manually giving feedback to them is very time consuming, and the feedback is not always precise or timely. In the era of Massive Open Online Courses (MOOCs), we ex-

L@S 2015, March 14–18, 2015, Vancouver, BC, Canada.

ACM 978-1-4503-3411-2/15/03...$15.00.
http://dx.doi.org/10.1145/2724660.2724682



(a) missing line (top view) (b) different scale (right view)

(c) misaligned right view (d) mirrored right view

Figure 3. Four typical cases of mistakes. Note that the labels on the views are not present in the actual multiview drawing.

pect high future demands for this type of engineering drawing course on an even larger scale, for which an automated grading/feedback tool would be critical. Particularly in a MOOC, but also with the large variety of backgrounds of students taking our on-campus course, different levels of students are engaged in the same curriculum. For effective education, we envision a system that should be able to distinguish them and provide specialized additional focused instruction and practice problems for different groups of students. To understand where students make mistakes frequently, an automated grading tool is essential not only for grading but also for analyzing big data.

Our autograder addresses several frequent error types that inexperienced engineers and designers make [11], summarized below.

Missing and Incorrect Lines

A common problem with hand-created or 2D Computer-Aided Design (CAD) software-created drawing is that one or more lines may be missing. Figure 3a shows this error type. This error is especially difficult to recognize when someone else made the drawing [1]; even when a grader has a solution to compare with, the grader may miss such a subtle mistake.

Mismatched View Scales

Each view of a drawing must have the same scale. Figure 3b shows an example when the scale of the right view is different, which makes for misaligned features between views. This is not permitted in multiview drawings. Note that as long as a drawing has the same scale throughout the views, the scale itself can be arbitrary for undimensioned drawings. So an automated grading tool should be *scale-invariant*, yet recognize mismatched scales between views in the same drawing.

Misaligned Views

Misaligned views, as shown in Figure 3c, also make it difficult for a human to match up features between adjacent views; they are not permitted in multiview drawings. The orthogonal views must be aligned both horizontally and vertically. Note that once the views are aligned appropriately, the offset distances between pairs of adjacent views do not need to match. So an automated grading tool should be *offset-invariant*. Moreover, because the entire drawing can be translated anywhere relative to the origin, the grading tool should be *translation-invariant*, up to alignment and relative location of views.

Views in Incorrect Relative Locations

Each view in a drawing must be located appropriately with respect to each other view. One possible mistake is caused by confusion of views (e.g., mistakenly placing a left view in the right view location). Sometimes students mistakenly rotate an entire view, typically by 90°. Another mistake is mirroring a view, as shown in Figure 3d.

These subtle mistakes are very easy for students to make, and are also easy for graders to miss. Especially with the traditional grading method where each student's printed drawing is graded by comparing it with a printed solution, a human grader can not guarantee a perfect comparison.

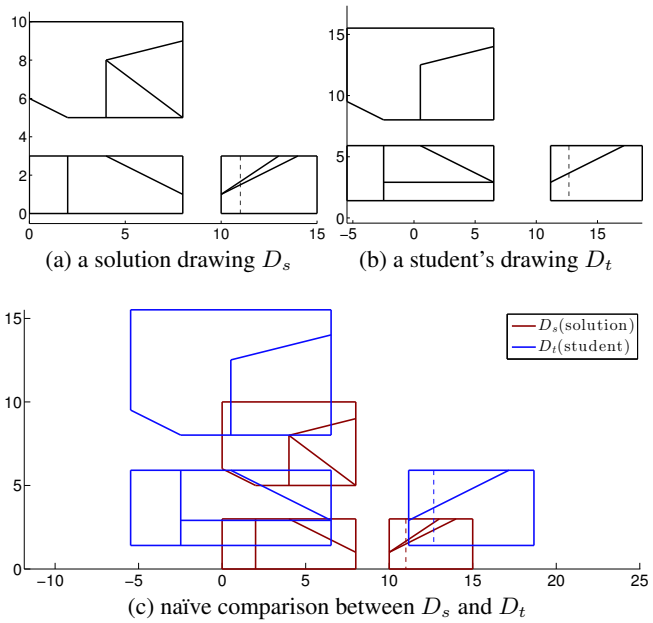
We show an example of a solution drawing and a student's drawing in Figure 4. Since they have different scale, translation, and offsets, the naïve comparison shown in Figure 4(c) does not work. Therefore we propose that an automated grading tool should be translation, scale, and offset-invariant when grading individual views, yet take these factors into account between views.

In this paper, we propose a simple and flexible automated grading/feedback system, which is translation, scale, and offset-invariant in the sense described above. The proposed algorithm determines the transformation information for each view (top, front, and right) in a drawing (Section "Algorithm"). We implement the automated grading/feedback system using MATLAB and address how the student errors detailed above can be graded using the transformation information (Section "Grading Checks").

RELATED WORK

To our knowledge, no existing work addresses machine grading of multiview engineering drawings. AutoCAD provides a plug-in called Drawing Compare [20], but it just visualizes the temporal changes of edits to a single drawing, and therefore it is not suitable to compare two drawings that include scale, translation, and offset differences.

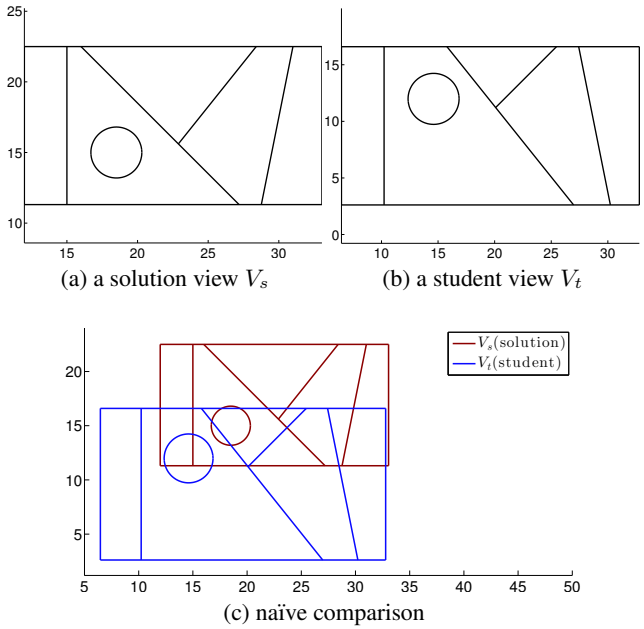
There has been research on multiview engineering drawing interpretation in the context of using the drawings as input to reconstruct 3D models [15, 18, 18, 6, 19, 10]. However, none



(a) a solution drawing D_s (b) a student's drawing D_t

(c) naïve comparison between D_s and D_t

Figure 4. An example of (a) a solution drawing and (b) a student's drawings and (c) their naïve comparison. Because they have different scales, translations, and offsets, a naïve comparison does not work.



(a) a solution view V_s (b) a student view V_t

(c) naïve comparison

Figure 5. An example pair of views for transformation estimation.

of these techniques are useful to compare and grade multiview drawings given that the reconstruction algorithms may fail when they face the incompleteness of students' drawings. Moreover the computation would be very intensive (both reconstruction and 3D object comparison).

On the educational side, Suh and McCasland developed education software to help train students in the interpretation of multiview drawings [16]. In their software, complete multiview drawings are given as input, and students are asked to draw the corresponding 3D models. This is very useful to enhance and evaluate students' multiview-drawings interpretation skills, the inverse of our purpose of evaluating students' multiview creation skills when 3D models are given as input. Since it is important for students to be familiar with popular CAD software such as AutoCAD, we chose to compare and grade native format AutoCAD files, which is easily extended to batch processing.

We use the random sample consensus (RANSAC) method [5] to estimate an affine transformation between the individual views of the two given drawings. RANSAC is an iterative method used to estimate parameters of a mathematical model from a set of data. RANSAC is very popular due to its effectiveness when the data has a high percentage of noise. The fact that much research in the computer vision field relies on RANSAC, for example, estimating the fundamental matrix [2], recognizing primitive shapes from point-clouds [14], or estimating an affine transformation between two line sets [4, 9], shows RANSAC's efficacy in multiple contexts. There have also been many variations introduced such as MLESACK [17] and Preemptive RANSAC [12], as well as research comparing the performance of the variations [3, 13]. In our current application, we have found that the original RANSAC concept is efficacious enough. We next discuss the basic RANSAC algorithm and how we apply it to estimate parameters of the transformation between single view drawings.

ALGORITHM

Single View Transformation Estimation

Initially we ignore the offset-invariant problem by assuming a drawing consists of only one view (e.g., front, top, or right). Let V_s be a single view from the source drawing (solution), and V_t be a single view from the target drawing (student's). Then the task here is to estimate the optimal transformation T^* between V_s and V_t in order to address the translation and scale-invariant problems. Once we know this transformation, we can transform V_s into the coordinate system of V_t. Let V_s' be the transformed version of V_s. We denote this as

$$T^* : V_s \rightarrow V_s'$$

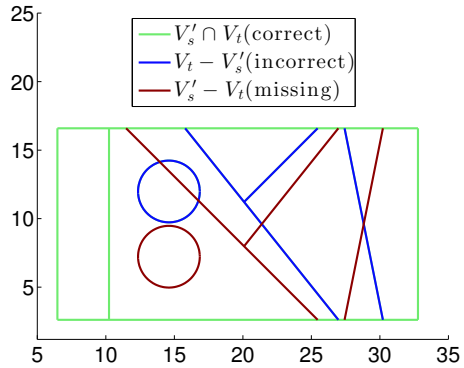
or equivalently,

$$V_s' = T^*(V_s).$$

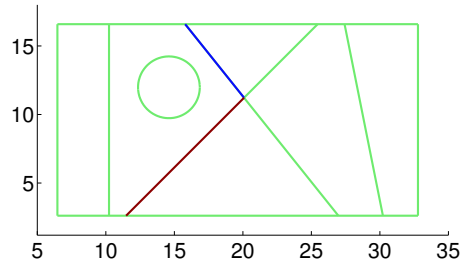
We can then compare V_s' and V_t fairly. (In the next section, we will discuss how to apply this single view transformation in the context of full multiview drawings to address the flexible offsets permitted between views.)

As a transformation model between the two views, we assume an affine transformation. Its parameters are translation in x and y (t_x, t_y), scale in x and y (s_x, s_y), rotation θ, and skew k.

We take the pair of drawings in Figure 5 as an illustrative example for this section. V_t (Figure 5(b)) was obtained from V_s (Figure 5(a)) by applying a uniform scale, mirroring, and



(a) no consideration of mirroring



(b) appropriate consideration of mirroring

Figure 6. (a) Even if we align the two views in terms of scale and translation, it is not easy to compare them at a glance; here half the elements still appear to be slightly off. (b) In fact, most elements match perfectly if the correct affine transformation is applied. The real problem is mirroring and two lines that only partially differ.

translation to V_s, and then editing two lines. It is not easy for a human to recognize what changes are there. The naïve comparison (Figure 5(c)) does not work at all. Even if scale and translation is properly considered, a grader may simply think most lines are slightly wrong as shown in Figure 6(a). However, better feedback can be provided by recognizing that the overall representation is in fact correct, except for mirroring and partially differing lines, as shown in Figure 6(b). Therefore, for a fair comparison that correctly identifies what conceptual errors led to the student's mistake, we need to estimate the affine transformation and use it to align the two drawings first, before comparing individual line elements.

The affine transformation estimation procedure is based on RANSAC, which consists of the following generically described four steps:

1. At each iteration, randomly select a subset $\mathcal{S}$ of the data set $\mathcal{D}$. Hypothesize that this subset (called *hypothetical inliers*) satisfies the ground truth model we seek.

2. Solve (or fit) the hypothetical model parameters Θ based on the hypothetical inliers $\mathcal{S}$. Note that $\mathcal{S}$ is the only input for choosing Θ, so if $\mathcal{S}$ includes incorrect of "noisy" elements, naturally the estimated model parameters Θ will not be high quality.

3. Evaluate the estimated model parameters Θ using all data $\mathcal{D}$. The subset $\mathcal{C} \subseteq \mathcal{D}$ whose members are consistent with the estimated model parameters Θ is called a *consensus set*.

4. Iterate steps 1-3. The optimal choice of model parameters Θ^* is that with the largest consensus set. Terminate when the probability of finding a better consensus set is lower than a certain threshold.

Our data set $\mathcal{D}$ is obtained by extracting certain points related to the elements in the drawings. The element types that we currently consider are *line*, *circle*, and *arc*. The point set consists of the two endpoints of line elements and the center points of circle and arc elements. Let P_s and P_t be the point sets extracted from all elements (lines, circles and arcs) from V_s and V_t, respectively. P_s and P_t together comprise the data set $\mathcal{D}$.

2D Affine transformations have six degrees of freedom (DOFs): two each for translation and scale, and one each for rotation and skew: therefore three noncolinear point pairs (correspondences between the two views) will give a unique solution. We randomly pick three ordered points from both P_s and P_t, and pair them in order. The three randomly selected point pairs are the hypothetical inliers $\mathcal{S}$, and we solve for the (hypothetical) affine transformation matrix T based on the three pairs of points. The full 3×3 affine transformation matrix can be solved for by using the homogeneous coordinate representation of the three pairs of points. (See for example [8] for more details.)

To evaluate T, we now transform the entire point set P_s by T. Let P_s' be the transformed version of P_s. If T is the optimal transformation, then most or even all points of P_s will be coincident with those of P_t. We define the consensus set $\mathcal{C}$ as $\mathcal{C} = P_s' \cap P_t$. Our evaluation metric is the cardinality of the consensus set (that is, the number of coincident points). We iterate this process; the optimal affine transformation T^* is the T with the largest $|\mathcal{C}|$. We can denote this as

$$\begin{aligned} P_s' &= T(P_s) \\ T^* &= \arg\max_T |\mathcal{C}| \\ &= \arg\max_T |P_s' \cap P_t)| \\ &= \arg\max_T |T(P_s) \cap P_t)| \end{aligned}$$

where $\arg\max$ stands for the argument of the maximum, the argument element(s) of the given argument space for which the given function attains its maximum value.

We terminate the iteration when $|C| > R * \min(|P_s|, |P_t|)$, where R is the minimum match rate, or all the cases are checked. We have found $R = 80\%$ to work well in practice. Once we have found a transformation that matches more than 80% of the points in the solution subview with points in the target drawing, we have found the region of interest that we are searching for, and there is no need to search further.

Consider the example of Figure 5; the optimal affine transformation is

$$T^* = \begin{bmatrix} 1.2494 & 0 & -8.5118 \\ 0 & -1.2494 & 30.7256 \\ 0 & 0 & 1 \end{bmatrix},$$

or equivalently, $t_x = -8.5118, t_y = 30.7256, s_x = 1.2494, s_y = -1.2494, \theta = 0$, and $k = 0$. Figure 6(b) shows the comparison between the transformed version of V_s, $V_s' = T^*(V_s)$, and V_t. In other words, we know that V_s should be scaled by 1.2494 and -1.2494 along the x and y axes respectively, and translated by (-8.5118, +30.7256) in order to compare it to V_t. The opposite signs for the x and y scales indicates mirroring. There is no skew or rotation.

Application to Multiview Drawings

In this section, we discuss how to apply the transformation estimation process to multiview drawing grading. Again, let the source drawing D_s be the solution drawing and the target drawing D_t be a student's drawing.

First a grader must manually subdivide the solution drawing (but not the student's drawing) into the front, right, and top views. Call them V_{front}, V_{right}, and V_{top}, respectively:

$$D_s = V_{front} \cup V_{right} \cup V_{top}.$$

In the general case, a view can be any subset of the solution drawing. One can specify arbitrary views V_i depending on the complexity of the solution drawing:

$$D_s \supseteq \bigcup_i V_i.$$

We individually estimate optimal transformations $T^*_{V_i}$ between each view V_i ($\subseteq D_s$) and the entire student drawing D_t. By calculating separate transformations for each view, we can address offset flexibility.

Consider the example input shown in Figure 4. For the front view, we have $t_x = -5.4785, t_y = 1.4114, s_x = 1.5, s_y = 1.5, \theta = 0$, and $k = 0$. For the top view, we have $t_x = -5.4785, t_y = 8.0145, s_x = 1.5, s_y = 1.5, \theta = 0$, and $k = 0$. For the right view, we have $t_x = 11.1657, t_y = 1.4114, s_x = 1.5, s_y = 1.5, \theta = 0$, and $k = 0$.

We next discuss how these components, and their relationships, can be used to grade the student drawing.

GRADING CHECKS

Once the optimal transformations $T^*_{V_i}$ (and their components) are calculated, one can set up a flexible set of rubrics. The checks described here correspond to the common student errors presented in the introduction.

Element Comparison

By applying each transformation $T^*_{V_i}$ to the corresponding view V_i from the solution D_s, we can compare individual elements of the two full multiview drawings. Suppose we have $T^*_{V_{front}}, T^*_{V_{top}}$ and $T^*_{V_{right}}$ from D_s. The transformed version D_s' is:

$$\begin{aligned} D_s' &= \bigcup_{V_i} T^*_{V_i}(V_i) \\ &= T^*_{V_{front}}(V_{front}) \cup T^*_{V_{top}}(V_{top}) \cup T^*_{V_{right}}(V_{right}). \end{aligned}$$

Figure 7 shows the transformed version of the solution, D_s', super imposed on the student's drawing D_t. The transformed version has the same location, offset and scales as the student's. In Figure 7(c), red highlights the missed elements (elements that exist in the solution, but not in the student's drawing: $D_s' - D_t$), and blue highlights the incorrect elements (elements that exist in the student's drawing, but not in the solution, $D_t - D_s'$). If both set differences are the empty sets, the two drawings are the same up to scale, translation, rotation, and skew.

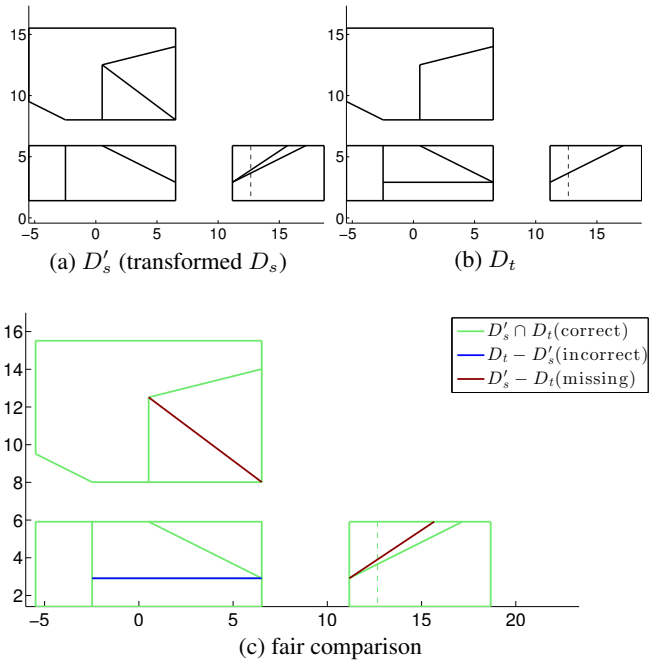


Figure 7. We estimate the transformation for each view individually using RANSAC. By applying the transformations to the views in D_s, we get the transformed version D_s'. Then the elements of D_s' and D_t can be compared one by one.

Front-Right View Alignment

The front view and right view should be aligned horizontally. This can be checked by confirming that the t_y components of $T^*_{V_{front}}$ and $T^*_{V_{right}}$ are the same. We also need to check if the right view is in fact to the right side of the front view in the student's drawing (in other words, t_x of $T^*_{V_{right}}$ should be greater than t_x of $T^*_{V_{front}}$.)

Front-Top View Alignment

The front view and top view should be aligned vertically. This can be checked by confirming that the t_x components of $T^*_{V_{front}}$ and $T^*_{V_{top}}$ are the same. We also need to check if the top view is on the upper side of the front view in the student's drawing (in other words, t_y of $T^*_{V_{top}}$ should be greater than t_y of $T^*_{V_{front}}$.)

Uniform Scale

In multiview drawings, the aspect ratio must be preserved, and all views must have the same scale, even though the scale factor itself can be arbitrarily. This can be checked

by confirming that all six scale components (s_x and s_y of $T^*_{V_{front}}, T^*_{V_{top}}$, and $T^*_{V_{right}}$) are the same.

Mirroring

By confirming that the signs of all six scale components are positive, we can recognize mirroring, which should not be present.

Rotation / Skew

The rotation and skew components of the transformations of all views should be zero, as long as the homework assignment is to reproduce the typical front, top, and right views.

COMPUTATION FILTERING

Suppose we estimate a transformation between point sets P_s and P_t. Let n_s and n_t be the cardinality of P_s and P_t, respectively. In the hypothesis generation step in RANSAC, there are $6\binom{n_s}{3}\binom{n_t}{3}$ possible cases, and for each case we need $n_s n_t$ comparisons to calculate the consensus set. This requires a huge number of iterations in the worst case. But we can filter out some hypotheses to reduce computation, as follows.

Choice Filtering

We store two simple attributes with each point: the element type ($\in \{line, circle, arc\}$) that gave rise to the point, and the number of intersecting elements at the point (in the case of the center points of circle and arcs, the number is zero). In the hypothesis generation step, we skip a hypothesis if these attributes of any of the pairs are inconsistent.

Transformation Filtering

Because the hypothesis transformations are acquired from the randomly chosen set of three pairs of points, most of them imply severe distortions, which are not typical of student errors. We can skip the evaluation step for this kind of unrealistic transformation. To filter out these cases, when we solve for T, we decompose T into its six components (DOFs). The unrealistic cases include when the absolute value of translations are too big, scales are too big, too small, or too imbalanced, etc. We skip those where:

- translation: $|t_x|, |t_y| > 300$;
- scale: $|s_x|, |s_y| < 1/3$ or $|s_x|, |s_y| > 3$;
- skew: $|k| > 0.1$; and
- rotation: no constraint.

Here the thresholds for t_x, t_y, s_x and s_y are practically determined based on the default visible canvas size when AutoCAD is opened. Students do not make mistakes such that the skew is nonzero, so theoretically k should be always zero, but due to numerical errors, the k value may be a very small number, e.g. 10^{-8}. Note that this is solely to reduce the search space, and one can shrink/expand the permissible ranges if analysis of a larger dataset indicates smaller/larger valid variations in students' drawings.

IMPLEMENTATION ISSUES

In practice, there are additional steps that needed to be implemented to fully automate the grading/feedback system. We briefly mention some of them below.

Converting DWG → DXF

Students draw using AutoCAD, which by default saves files in DWG format. Because AutoCAD is commercial software and DWG is its binary file format, to our knowledge, there is no open source code for accessing DWG files directly. So we need to convert DWG files to DXF file format, which is a file format developed by AutoDesk for enabling data interoperability between AutoCAD and other programs. For an automated batch process of this conversion on all students' submissions, we also implemented an AUTOLISP script, which runs in AutoCAD.

Loading DXF in MATLAB

We extract drawing elements from each DXF file using MATLAB. Currently the loading operation is based on the open source code in the MATLAB CENTRAL File Exchange website [22, 21].

Merging Elements

Some elements may be drawn (partially) duplicated, overlapping, or decomposed into sub-segments. Especially in the case of lines/arcs, one may have several connected or overlapping partial lines/arcs instead of one long line/arc. For this reason, we merge objects into one if they can be represented as a simpler one. This also makes the point set smaller, which reduces computation time.

Pre-defining Layer Names

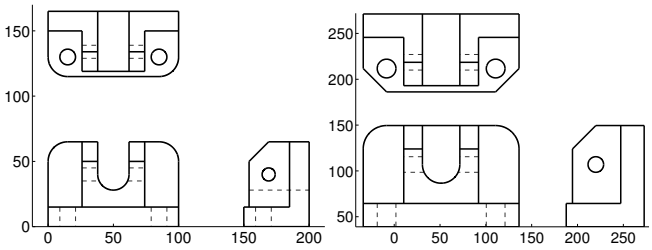
Currently we do not autograde dimensioning and header parts of multiview drawings, only visible and hidden lines. Since visible and hidden lines should be drawn with different line styles and thickness, we teach students to put them in separate layers and define these properties to apply to the entire layer. For autograding, we provide a template with the layer names, and only load elements drawn on the visible and hidden layers. Even though giving predetermined layer names is a constraint for the autograding system, declaring layers and grouping objects of the same type into a single layer is an important skill for student to learn regardless.

RESULTS

We show another grading example in Figure 8. The solution drawing D_s (Figure 8a) and the student drawing D_t (Figure 8b) can not be compared using a naïve algorithm due to translation, scale, and offsets (Figure 8c). Using the estimated transformation for each view, we take our transformed version of the solution, D'_s, and compare D_s and D_t (Figure 8d).

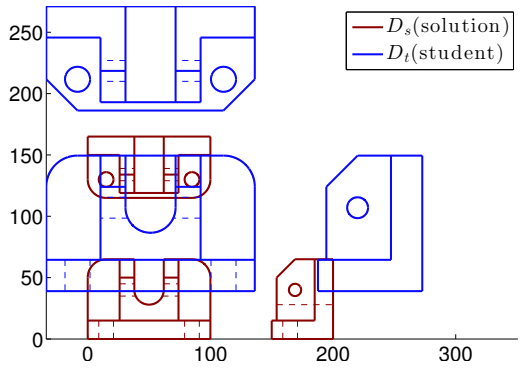
Grading result visualization

Beyond the advantages of more accurate, timely feedback to students, another advantage of an autograding tool will be its ability to analyze and summarize the grading results. As an example, we can visualize which elements of a drawing were most frequently drawn incorrectly by students, which can be useful information for instructors. We ran our algorithm in batch mode on the submissions in Fall 2013 for two problems assigned in homeworks #2 and #3 in E28. These assignment batches consisted of 115 and 113 students' submissions

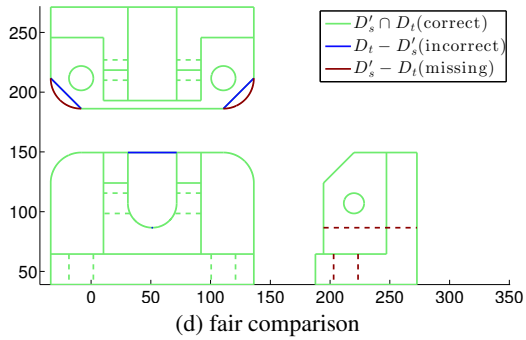


(a) solution D_s

(b) student D_t



(c) naïve comparison



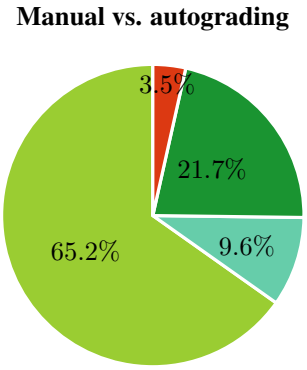
(d) fair comparison

Figure 8. Comparing the solution to a student's drawing. To be compared to D_t, all views in D_s are scaled 1.7 times larger. The top, front, and right views are translated (-33.8, +186.2), (-33.8, +39), and (+187.7, +39), respectively. All views have zero rotation and skew. By aligning them, the algorithm finds incorrect and missing lines, which are represented in blue and dark red.

respectively. For each element in the solution drawing, we count in how many student submissions it is "missing." Similarly, for each "incorrect" element in the student drawing, we count how many student submissions have it. Figure 9 shows the solution with the elements color coded: the most difficult elements — that are most frequently missing/incorrect — are represented in dark red/blue, and those less frequently missing/incorrect are represented in light red/blue. In the problem from assignment #2, we can see that the top view causes more mistakes than the other views, and that students miss the diagonal and hidden lines in the front and right view most frequently (Figure 9a). Figure 9b shows that the diagonal features are frequently misdrawn. In the problem from assignment #3, many times students get confused in the upper part of the front view, and hidden lines are frequently missed.

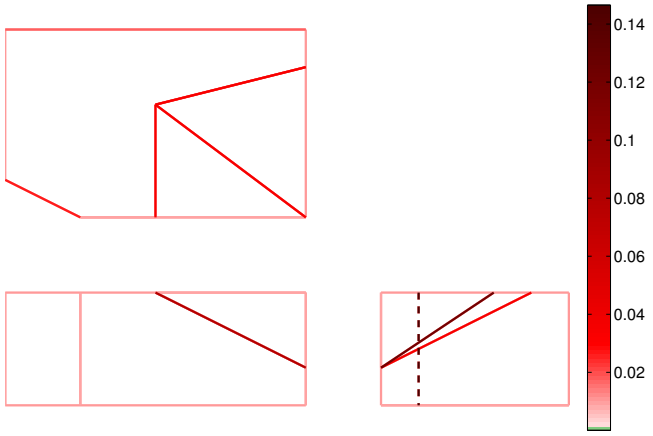
Comparison with human grading

To verify the efficacy of the proposed algorithm, we compare the autograding results with human grading. A human grader with a full semester of experience grading for the course graded the 115 submissions of the homework #2 problem introduced above, using gradescope [7] with pdf files of the submission.

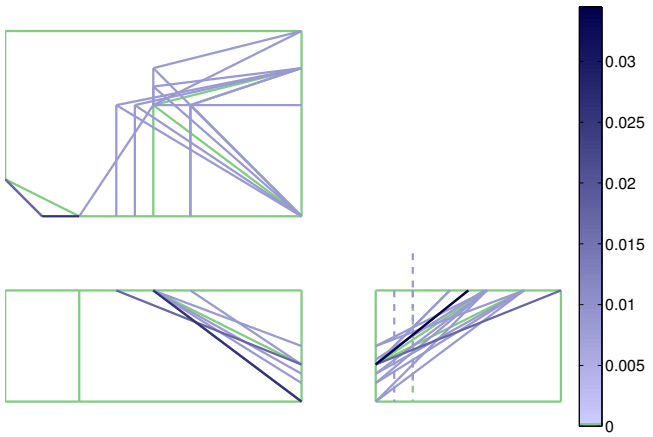


- Category A: Same feedback.
  Both give the same feedback.
- Category B: Similar feedback.
  The same errors are identified, but described differently.
- Category C: Better autograding feedback.
  Manual grading fails to catch some mistakes.
- Category D: Incorrect autograding feedback.
  Autograding fails to estimate proper transformation.

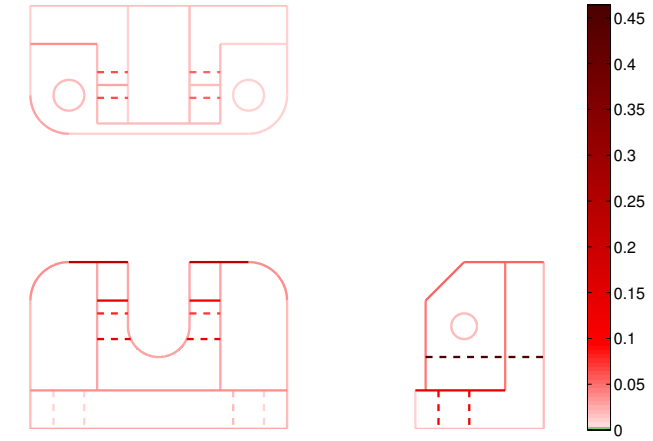
We divide the comparison results into four categories. In the case of category A and B, autograding and human grading find the same errors, which account for 74.8% of the total 115 submissions. In the case of category B, although the same drawing elements are identified as errors, the human grader described them differently in her grading feedback to the student. Figure 10 shows two examples of category B. While the human grader interprets the mistake as "lines not aligned," the autograder reports it as the number of missing lines and incorrect lines. The human's interpretation can be more flexible,



(a) visualization of missing errors on a problem from assignment #2

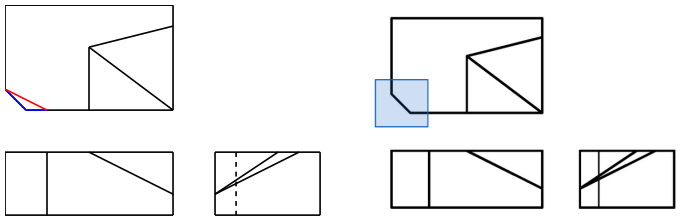


(b) visualization of incorrect errors on a problem from assignment #2. Correct lines (solution) are shown in green.

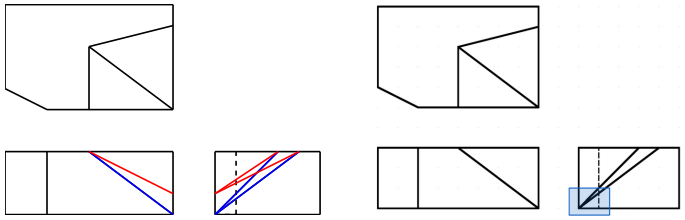


(c) visualization of missing errors on a problem from assignment #3

Figure 9. Color coded difficulty. The elements that are most frequently "missing" are shown in dark red, and those less frequently missing are shown in light red (a and c). The elements that are most frequently "incorrect" are shown in dark blue, and those less frequently incorrect shown in light blue (b). The numbers in the color bar indicate the fraction of student submissions that made the mistake for each element.



(a) While autograding (left) reports "1 missing line, 2 incorrect lines", a human grader (right) reports "1 incorrect position."



(b) While autograding (left) reports "3 missing line, 3 incorrect lines", a human grader (right) reports "1 line not aligned."

Figure 10. Two examples of category B. Even though different rubrics are applied, errors are identified.

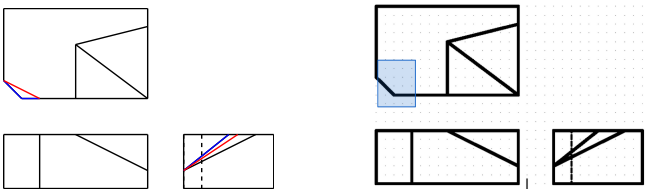
nuanced, and higher level. We leave more advanced emulation of such human grading rubrics as future work.

The most dramatic result is category C. For 21.7% of the submissions, the new autograding system catches students' mistakes that the human grader misses. We show two examples in Figure 11. This happens especially when a drawing includes subtle mistakes such as a slightly incorrect location, and/or when a drawing includes incorrect locations that are nonetheless consistent in neighboring views. In these cases, human graders may not notice them on the printed drawing.

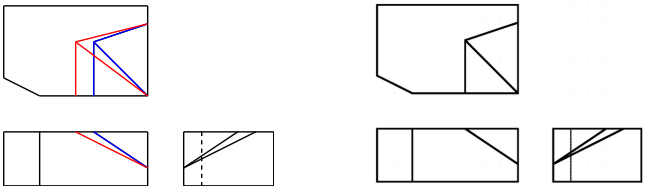
For 3.5% of the submissions, the autograding system failed to estimate the appropriate transformation, and gave incorrect feedback. We show an example in Figure 12. The student drawing (Figure 12b) has an incorrect front view. Note that our RANSAC evaluation metric is the number of coincident points. When a student drawing has several wrong elements, the RANSAC algorithm may regard a strange transformation as the best transformation for the reason that it yields the maximum number of coincident points. We expect that this problem can be solved by extending the RANSAC evaluation metric to consider lines as well as points in future work. Note that even though the proposed algorithm fails to give correct feedback for 3.5% of the submissions, this happens only when the student drawing has multiple errors. In these cases, a whole view was reported as incorrect by the autograder, where in fact some partial credit should have been given. The proposed algorithm never failed to recognize perfect drawings as such.

CONCLUSION

Motivated by the importance of an automated grading system for multiview drawings, we propose a novel system that can compare two multiview drawings, a reference solution and a student's submission. These drawings may have inconsistent translations, scales, mirroring, skew, and/or rotation, all of

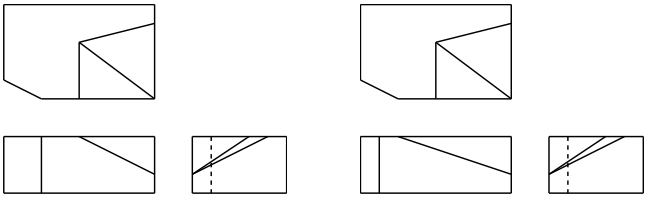


(a) A human grader (right) succeeded in recognizing the incorrect elements in the top view, but failed to catch the element in a slightly incorrect position in the right view.



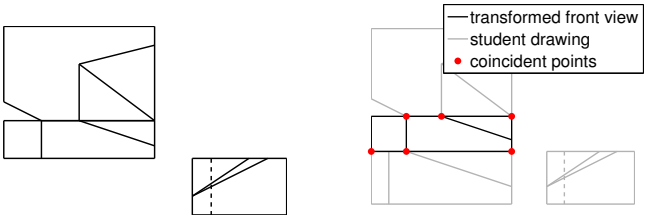
(b) A human grader (right) failed to notice elements in incorrect positions and gave a perfect score. Such errors are difficult for humans to catch because the positions are consistent with the neighboring view.

Figure 11. Two examples of category C. While a human grader failed to notice these mistakes, our autograding system found them.



(a) solution drawing

(b) student drawing



(c) transformed solution. The transformation estimate for the front view is wrong.

(d) The wrong transformation is chosen because it matches six points among the total eight points of the front view.

Figure 12. An example of category D. Our algorithm failed to estimate an appropriate transformation for the front view of the solution drawing. Note that the student drawing (b) has multiple mistakes in the front view.

which must be distinguished from allowable differences in scale, offset, and translation to reliably identify and classify errors in the students' drawings.

Our system provides fair comparison and grading checks for students' drawings, which can be used as input for a flexible scoring system. For example, in many cases, a grader may not want to reduce scores for duplicate errors of the same type. A grader may want to place different scores (emphasis) on different elements. One possibility would be to use our element-wise difficulty analysis to assign an appropriate score to each element.

The proposed system can be useful for large classes, eliminating time consuming manual grading and incomplete feedback, and for MOOCs on engineering drawing, which currently do not exist.

ACKNOWLEDGMENTS

We thank Zhongyin Hu for performing the manual grading for comparison. We also thank Armando Fox for offering an inspiring course on online education and autograding. We also thank many E28 students who tested submission instructions for autograding and whose assignments provided us with test data, and Anastasia Shuller for beta testing the autograder. This work was funded in part by UC Berkeley's vice provost for teaching and learning.

REFERENCES

1. Bertoline, G., Wiebe, E., Hartman, N., and Ross, W. *Technical Graphics Communication*. McGraw-Hill Science/Engineering/Math, 2002.
2. Brown, M., and Lowe, D. G. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Fifth International Conference on 3-D Digital Imaging and Modeling* (2005), 56–63.
3. Choi, S., Kim, T., and Yu, W. Performance evaluation of RANSAC family. *Journal of Computer Vision* (1997).
4. Coiras, E., Santamar, J., and Miravet, C. Segment-based registration technique for visual-infrared images. *Optical Engineering 39*, 1 (2000), 282–289.
5. Fischler, M. A., and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (June 1981), 381–395.
6. Geng, W., Wang, J., and Zhang, Y. Embedding visual cognition in 3D reconstruction from multi-view engineering drawings. *Computer-Aided Design* (2002).
7. gradescope.com. `https://gradescope.com`, Last accessed on Jan 16, 2015.
8. Hartley, R., and Zisserman, A. Multiple View Geometry in Computer Vision, 2003.
9. Kwon, Y. P. Line segment-based aerial image registration. Master's thesis, EECS Department, University of California, Berkeley, May 2014.

10. Lee, H., and Han, S. Reconstruction of 3D interacting solids of revolution from 2D orthographic views. *Computer-Aided Design 37*, 13 (Nov. 2005), 1388–1398.

11. Lieu, D., and Sorby, S. *Visualization, Modeling, and Graphics for Engineering Design*, 3rd ed. Delmar Learning, 2009. Chapter 10.

12. Nistér, D. Preemptive RANSAC for live structure and motion estimation. *Machine Vision and Applications* (2005).

13. Raguram, R., Frahm, J.-M., and Pollefeys, M. A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In *ECCV 2008*. 2008, 500–513.

14. Schnabel, R., Wahl, R., and Klein, R. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* (2007).

15. Shin, B. S., and Shin, Y. G. Fast 3D solid model reconstruction from orthographic views. *Computer-Aided Design 30*, 1 (Jan. 1998), 63–76.

16. Suh, Y. S., and McCasland, J. Interactive Construction of Solids from Orthographic Multiviews for an Educational Software Tool. *Computer-Aided Design and Applications 6*, 2 (Jan. 2009), 219–229.

17. Torr, P. H. S., and Zisserman, A. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding 78*, 1 (Apr. 2000), 138–156.

18. Wang, W., and Grinstein, G. G. A survey of 3D solid reconstruction from 2D projection line drawings. *Computer Graphics Forum 12*, 2 (May 1993), 137–158.

19. Wang, Z., and Latif, M. Reconstruction of a 3D solid model from orthographic projections. In *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings* (2003), 75–82.

20. AutoCAD Drawing Compare Plug-in. **`https://apps.exchange.autodesk.com/VLTC/en/Detail/Index?id=appstore.exchange.autodesk.com%3Adrawingcompare%3Aen`**, Last accessed on Oct 20, 2014.

21. MATLAB CENTRAL, File Exchange. **`http://www.mathworks.com/matlabcentral/fileexchange`**, Last accessed on Oct 20, 2014.

22. Read DXF File Data. **`http://www.mathworks.com/matlabcentral/fileexchange/24572-read-dxf-file-data`**, Last accessed on Oct 20, 2014.