Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

OptiComics: A Mathematical Programming Framework for Geometry-Preserving Comic Lettering

Alessandro Bicciato
Ca' Foscari University of Venice
Giovanni Fasano
fasano@unive.it

Ca' Foscari University of Venice

Case Report

Keywords:

Posted Date: October 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11223851/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: Competing interest reported. Alessandro Bicciato declares no relevant financial or non-financial interests. Giovanni Fasano is an Associate Editor of Operations Research Forum and is not involved in the editorial handling of this manuscript.

`OptiComics`: A Mathematical Programming Framework for Geometry-Preserving Comic Lettering

Alessandro Bicciato[1] and Giovanni Fasano[2*]

[1] Department of Environmental Sciences, Informatics and Statistics, Ca' Foscari University, Scientific Campus, via Torino 155, Venice (Mestre), 30172, Italy.

[2*] Venice School of Management, Ca' Foscari University, San Giobbe, Cannaregio 813A, Venice, 30121, Italy.

*Corresponding author(s). E-mail(s): fasano@unive.it;
Contributing authors: alessandro.bicciato@unive.it;

Abstract

Automatic lettering of translated comics is a constrained layout problem in which a fixed visual composition must accommodate target-language text with different lengths, hyphenation patterns, and writing conventions. We present the optimization core of `OptiComics`, a framework that separates this task into a continuous geometric stage and a discrete typographic stage. A sampled speech-balloon contour is first summarized by Principal Component Analysis and refined through a nonconvex nonlinear program that selects a large text-safe ellipse while keeping the artist's balloon geometry unchanged. The resulting ellipse is then filled with text, and an integer linear programming model assigns ordered syllables to text rows, while accounting for spaces, hyphenation, font size, centering, and visual balance. We provide detailed derivations, worked numerical examples, modeling clarifications concerning rotated coordinates and sample-based containment, and implementation guidance. The framework is evaluated on real pages from Japanese and Italian comics, including highly irregular balloons and full-page production workflows. The results show that mathematical programming can be embedded effectively in an editorial production pipeline for the comics industry, while also identifying important limitations associated with irregular geometry, composite balloons, raster discretization, and human-facing typographic quality.

Keywords: Operations Research, Comics Lettering, Nonlinear Programming, Integer Linear Programming, Principal Component Analysis

Mathematics Subject Classification (MSC): 90C30, 90C11

1 Introduction

Comics translation is an unusually constrained publishing task because verbal content is embedded in a visual composition that is normally treated as fixed. A translated sentence can be longer or shorter for different languages, hyphenation similarly changes with language, and the source language may use a writing direction (e.g., vertical writing) that differs from

the target language (e.g., left-to-right writing). Yet panels, balloons, tails, drawings, and page order should generally remain faithful to the artist's design. The practical problem is therefore not merely translation but *geometry-preserving lettering*: after an approved target-language sentence has been produced, one must determine where it can be placed, at what font size, and how it should be broken into lines without colliding with the balloon boundary or producing an unattractive layout.

These constraints are well known in comics and translation studies, including work on visual language, cultural production, translation for younger audiences, and translation as a socially situated practice [1–4]. As a more general issue, comics combine verbal, visual, spatial, and graphic systems [5–9]. Translation may require adaptation of cultural references, sound effects, onomatopoeia, names, catchphrases, and other multimodal elements [10–14]. Such semantic choices remain outside the scope of the optimization models developed here: `OptiComics` assumes that the target text for each balloon has already been supplied by a translator or another authorized process. The optimization task starts at the lettering stage.

The computational-comics literature has addressed complementary problems such as panel and text extraction, speech-balloon detection and segmentation, character recognition, OCR, and benchmark construction [15–23]. Recent systems also explore machine translation and multimodal narrative processing [24, 25]. Our focus is different: given a prepared balloon mask and target sentence, we formulate the placement problem explicitly as mathematical programming. This makes feasibility conditions and aesthetic surrogates inspectable and allows the editorial workflow to retain hard geometric constraints rather than relying only on generative prompting.

`OptiComics` decomposes lettering into two coupled optimization stages. First, the contour of a balloon is sampled and summarized through Principal Component Analysis (PCA) [26, 27]. A nonconvex nonlinear program then refines the center and semiaxes of a large ellipse whose orientation is fixed by the PCA estimate. The ellipse is a conservative surrogate for the useful text region: it is simple enough to support robust row-width calculations while respecting the original balloon geometry. Second, an Integer Linear Programming (ILP) model assigns ordered syllables to rows. The discrete formulation accounts for row capacities, blank spaces, hyphenation, central placement, and a preference for visually balanced line lengths. Standard nonlinear- and integer-programming concepts therefore enter the same editorial pipeline [28–32].

Three points are important for interpreting the contribution. First, the nonlinear model (see also [33]) that computes the ellipse is nonconvex and its boundary conditions are imposed on a finite set of sampled contour points. Hence, the computed ellipse is a *large sample-consistent local solution*, not a certificate of the globally largest ellipse inside the exact continuous balloon. Second, the method is intentionally modular: improved segmentation, OCR, translation, or graphics interfaces can replace upstream or downstream components without changing the mathematical core. Third, the objective is not to automate cultural judgment. Layout automation is useful precisely when it removes repeated geometric work while keeping the translation, final typography, and quality control editable by human practitioners.

The numerical study uses real Japanese and Italian comics and includes irregular balloons, different reading directions, full-page workflows, and chapter-scale timing. On a 15-page chapter containing 162 balloons, the complete process takes 346.757 seconds on the reference machine, with optimization and graphics-software placement contributing almost equal shares of total time. Besides the main formulations, we provide worked examples and implementation explanations intended to make the method reproducible and to clarify where the optimization assumptions matter in practice.

To the best of the authors' knowledge, `OptiComics` is one of the first optimization frameworks specifically designed to preserve pre-existing speech-balloon geometry, while combining *continuous optimization of a text-safe region with an integer model for ordered, hyphenation-aware text allocation* (see also [34–36]).

The remainder of the paper is organized as follows. Section 2 summarizes the technical characteristics of comics pages and related computational approaches. Section 3 reports some related approaches from the literature. Section 4 develops the geometric and integer models. Section 5 gives derivations and short worked examples. Section 6 describes the implementation workflow, and Section 7 reports the page- and chapter-level experiments. Section 8 discusses limitations and future research.

2 Technical characteristics of comics pages and balloon segmentation

A comics page is a structured combination of drawings, panels, balloons, captions, text, and graphical marks. For automatic lettering, several tasks that are conceptually distinct must nevertheless be coordinated: (i) identifying panels and text regions; (ii) distinguishing speech balloons from other graphic elements; (iii) extracting a sufficiently accurate contour; (iv) preserving the association between text and the corresponding character; (v) maintaining reading order; and (vi) placing the translated text without altering the artwork. These issues have motivated a broad literature on computational analysis of comics [15, 19, 37].

Panel and text extraction can be approached through connected components, topology, and spatial relations [16]. Speech-balloon detection has also been treated through active contours and text-independent segmentation [17, 18]. Manga analysis has increasingly used convolutional and deep neural networks for faces and other graphical objects [20, 21], while OCR pipelines combine text spotting and recognition with document-analysis methods [22]. These components are mostly complementary to the present work: `OptiComics` starts from a prepared balloon layer and addresses the downstream geometric and typographic allocation problem.

The geometry of the balloon is not a neutral container; scholarship on comics form and graphic literature also emphasizes the expressive role of such visual conventions [38]. Its contour, tail, position, and size contribute to the page composition and often to expressive meaning. A localization system should therefore avoid resizing the balloon merely because the target sentence is longer. This requirement is particularly strong when translating between writing systems. Japanese manga commonly combines vertical source text with a right-to-left page sequence, whereas English target text is usually horizontal; European comics may already use left-to-right navigation but still present very different balloon shapes. A robust lettering method must treat writing direction as input data rather than as a reason to redesign the page.

From an optimization viewpoint, the central geometric difficulty is to estimate a useful text region for arbitrary rasterized contours. Exact maximal-inscribed-shape problems can become expensive or brittle when boundaries are highly irregular. We use an ellipse because it offers a useful compromise: its orientation and scale can be initialized from contour statistics, its geometry is easily scalable, and horizontal row capacities can be obtained analytically. The price of this simplicity is conservatism for concave or composite balloons. Figures 5 and 6 later show two practical exceptions: a single ellipse can waste space in a highly irregular balloon, while small thought bubbles may be graphic signs that should not receive text at all.

The typographic difficulty is discrete. Once the text-safe region is known, syllables or hyphenation units must be assigned to consecutive rows. A row has a geometric capacity determined by the ellipse, but its used length also depends on the font, spaces, and whether a word is broken at the end of the row. The chosen breakpoints affect balance and readability. This resembles classical paragraph line breaking [39], but here the row capacities vary with vertical position because they are chords of an ellipse rather than constant-width columns. The resulting model is naturally expressed through binary assignment and transition variables.

3 Related computational approaches

Computer-vision and machine-learning methods continue to improve the upstream understanding of comics. Boundary-aware manga segmentation [40], standard border-following methods [41], corpus construction [19, 37], and recent surveys [23] provide increasingly strong tools for recognizing page components. More recent work has investigated automatic translation with visual reconstruction [24] and multimodal narrative description [25]. The distinctive feature of the present approach is the explicit mathematical-programming layer between recognition/translation and final graphics placement. Hard geometric constraints, line capacities, and layout penalties are duly represented, which makes the behavior of the method interpretable and permits editorial constraints to be modified without retraining a model. Finally, for some related approaches in the literature on balloon lettering, partially involving optimization procedures, we can consider [34–36].

4 Our proposal

This section describes the optimization core of `OptiComics` from a mathematical programming perspective. The workflow is designed to accept translated text from any source language and to place it in a target language that can be rendered with the available fonts and hyphenation rules. It is especially useful when source and target writing directions differ. The system does not replace the translator; rather, it automates the geometric and typographic decisions that follow translation. The first optimization stage identifies a text-safe region inside each prepared balloon or panel, and the second determines a readable allocation of the target text to rows while preserving the outer geometry of the page.

4.1 Identifying a large admissible text region

The first step in our automatic lettering procedure is to decide the precise area for the text placement. We scan each balloon so that we can extract the layer's pixels in RGB mode to assess the color. The color is picked from the center to recognize the correct encoding for the text styles. Then, the image is converted to black and white, applying a threshold when handling the RGB scan. This is done in order to have the overall balloon area in black and the surrounding area inside the bounding box in white (see Figure 1, where the rectangle represents the bounding box). The next step consists of extracting the contour of the balloon,

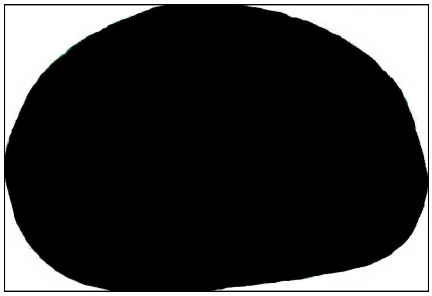


Fig. 1: Balloon area is represented using black color and with a white background in the corresponding bounding box.

which we perform using standard OpenCV methods (see *e.g.* https://opencv.org). To control model size, we retain a sufficiently accurate subset of contour samples. Consecutive retained points (x_1, y_1) and (x_2, y_2) satisfy $\max\{|x_1 - x_2|, |y_2 - y_1|\} \geq \epsilon > 0$, where all coordinates are expressed with respect to an origin in the bounding box. Horizontal scan lines are spaced according to the box height so that the contour is represented without unnecessary

oversampling. This reduces the computational cost of boundary extraction[1] of the balloon, while retaining the main geometric features of the boundary. The resulting point cloud is the input to the PCA initialization and nonlinear refinement described next.

4.2 PCA initialization of ellipse orientation and scale

Let the sampled balloon contour be represented by N points $p_h = (P_{hx}, P_{hy})^T$, $h = 1, \ldots, N$, in the pixel coordinate system of the balloon bounding box. The first purpose of PCA is to estimate an *orientation* for the ellipse semiaxes; the second is to provide a *scale* for the initial semiaxes. PCA is used only as an initializer: it does not certify that the final resulting ellipse is contained in the balloon. The sample centroid is

$$\bar{p} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} = \frac{1}{N} \sum_{h=1}^{N} p_h, \tag{1}$$

with coordinate variances

$$\sigma_x^2 = \frac{1}{N} \sum_{h=1}^{N} (P_{hx} - x_c)^2, \tag{2}$$

$$\sigma_y^2 = \frac{1}{N} \sum_{h=1}^{N} (P_{hy} - y_c)^2. \tag{3}$$

The so-called *covariance matrix* $\Sigma \in I\!R^{2\times 2}$ in the original pixel units is

$$\Sigma = \frac{1}{N} \sum_{h=1}^{N} (p_h - \bar{p})(p_h - \bar{p})^T = \begin{pmatrix} \sigma_x^2 & b \\ b & \sigma_y^2 \end{pmatrix}, \qquad b = \frac{1}{N} \sum_{h=1}^{N} (P_{hx} - x_c)(P_{hy} - y_c). \tag{4}$$

Using pixel coordinates here is important: the eigenvalues of Σ have units of pixels squared, so their square roots provide dimensionally meaningful scale indicators. Standardizing both coordinates can still be useful when one wishes to study correlation rather than physical scale, but the resulting eigenvalues should not be interpreted directly as semiaxes in pixels [26, 27].

Let (λ_1, V_1) and (λ_2, V_2) be the eigenpairs of Σ, ordered so that[2] $\lambda_1 \geq \lambda_2 > 0$ and $\|V_i\|_2 = 1$. The orthogonal matrix

$$R = (V_1 \; V_2) \tag{5}$$

defines coordinate axes aligned with the principal directions of the sampled contour. If $V_1 = (v_{11}, v_{21})^T$, the ellipse orientation (with respect to the horizontal axis) may be written as

$$\theta = \texttt{atan2}\left(\frac{v_{21}}{v_{11}}\right), \tag{6}$$

with the usual equivalence of directions modulo π. The source implementation additionally applies a readability rule: very small rotations are reset to horizontal, because a slight arbitrary tilt can impair reading while changing the usable area only marginally [42, 43].

Then, for initialization of the ellipse semiaxes we use

$$(r_x)_{init} = \sqrt{\lambda_1}, \qquad (r_y)_{init} = \sqrt{\lambda_2}, \tag{7}$$

as scale estimates. These quantities measure directional spread rather than exact boundary radii. Their only role is to start and bound the subsequent nonlinear refinement. In principle,

[1] We omit details for cases in which the contour of a balloon may be irregularly dotted/dashed.
[2] Our final implementation explicitly verifies that λ_1 and λ_2 are both positive.

alternative scale initializers, such as extrema of the projections $V_i^T(p_h - \bar{p})$, can be substituted without changing the optimization model.

For any candidate center (x_c, y_c) and the fixed angle θ, define the aligned coordinates of contour point h by

$$u_h(x_c, y_c) = \cos\theta \cdot (P_{hx} - x_c) + \sin\theta \cdot (P_{hy} - y_c), \tag{8}$$

$$v_h(x_c, y_c) = -\sin\theta \cdot (P_{hx} - x_c) + \cos\theta \cdot (P_{hy} - y_c). \tag{9}$$

A point is outside or on the ellipse with semiaxes r_x, r_y precisely when

$$\frac{u_h(x_c, y_c)^2}{r_x^2} + \frac{v_h(x_c, y_c)^2}{r_y^2} \geq 1. \tag{10}$$

This rotated-coordinate form is essential whenever $\theta \neq 0$. It also clarifies the information passed from PCA to nonlinear programming: PCA supplies the center, orientation, and scale; the subsequent nonlinear model adjusts the center and semiaxes while holding the orientation fixed.

4.3 A nonlinear model to compute a large sample-consistent inscribed ellipse

We next refine the PCA initialization to obtain a large text-safe region. For a balloon with sampled contour $\{(P_{hx}, P_{hy})\}_{h=1}^N$, equations (1), (6), and (7) define a preliminary ellipse $\mathcal{C}_0$ (Figure 2). This ellipse is only a geometric summary: its PCA scale need not be contained in the balloon and need not maximize usable area. Its centroid does, however, belong to the convex hull of the contour samples because it is their convex combination with weights $1/N$. The nonlinear stage therefore preserves a controlled neighborhood of this initialization while enforcing sample-based exclusion of the boundary points. Hence, we need a procedure

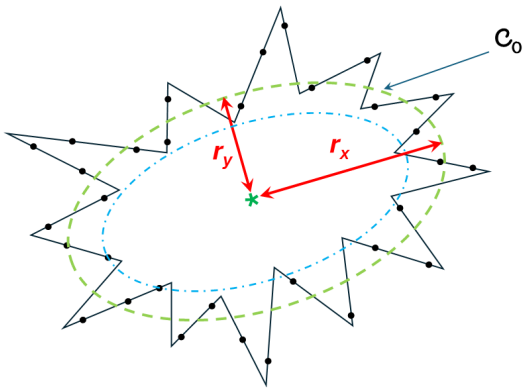


Fig. 2: The (dashed) larger ellipse $\mathcal{C}_0$ computed through the covariance matrix Σ in Section 4.2 possibly contains some points (black dots) on the contour of the *explosion* balloon. Conversely, the (dashed–dotted) smaller ellipse represents the largest ellipse which maintains the same center of $\mathcal{C}_0$ and *does not contain any sampled point* on the contour.

to guarantee that the initial computed ellipse $\mathcal{C}_0$, whose semiaxes are given in (7) and the rotation with respect to the bounding box axes is detailed in (6), can be modified, to obtain a large sample-consistent ellipse inside the current balloon. Because the contour is sampled

finitely and the model is nonconvex, the solution is not claimed to be a certified global maximum for the continuous raster region. This last task is pursued through the solution of a nonconvex nonlinearly constrained optimization model where:

- the angle θ is left unchanged (with respect to the computation in (6)), so that we use the PCA direction as a reliable orientation estimate and avoid adding another nonconvex decision variable;
- the initial center coordinates, as well as the (positive) semiaxes values of $\mathcal{C}_0$ computed using the covariance matrix, are provided to the nonconvex nonlinearly constrained optimization model as initialization values for variables, *i.e.*,

$$((x_c)_{init}, (y_c)_{init}) \quad \longleftarrow \quad (x_c, y_c),$$

$$((r_x)_{init}, (r_y)_{init}) \quad \longleftarrow \quad (r_x, r_y).$$

Then, the solution of the nonlinear model refines the above four variables in order to yield a maximally extended text-safe ellipse area, considering the angle θ as fixed. The rotation θ is essential when $\theta \neq 0$: the ellipse inequality must be evaluated along its principal axes rather than along the sides of the bounding box. We also stress that the finite set of contour constraints below certifies exclusion of the sampled points; for an arbitrary nonconvex raster contour it does not constitute a global certificate of containment at every unsampled boundary pixel. For this reason, here we use the term *sample-consistent text-safe ellipse* when a strict geometric qualification is relevant.

Overall, the nonlinear model (see also the classic reference [29] for basics on nonlinear programming) we solve is the following (min_A and max_A are proper positive values):

$$\max_{x_c, y_c, \alpha, r_x, r_y} r_x r_y \tag{11}$$

$$0 \leq \alpha_h \leq 1, \qquad h = 1, \ldots, N, \tag{12}$$

$$\sum_{h=1}^{N} \alpha_h = 1, \tag{13}$$

$$x_c = \sum_{h=1}^{N} \alpha_h P_{hx}, \tag{14}$$

$$y_c = \sum_{h=1}^{N} \alpha_h P_{hy}, \tag{15}$$

$$min_A \leq \pi r_x r_y \leq max_A, \tag{16}$$

$$\frac{u_h(x_c, y_c)^2}{r_x^2} + \frac{v_h(x_c, y_c)^2}{r_y^2} \geq 1, \qquad h = 1, \ldots, N, \tag{17}$$

$$\gamma (r_x)_{init} \leq r_x \leq \frac{\mathcal{M}}{\xi}, \tag{18}$$

$$\gamma (r_y)_{init} \leq r_y \leq \frac{\mathcal{M}}{\xi}, \tag{19}$$

$$(x_c)_{init} - \gamma\mathcal{M} \leq x_c \leq (x_c)_{init} + \gamma\mathcal{M}, \tag{20}$$

$$(y_c)_{init} - \gamma\mathcal{M} \leq y_c \leq (y_c)_{init} + \gamma\mathcal{M}, \tag{21}$$

and the (feasible) initial values for the unknowns we adopted are (see also (1))

$$\begin{cases} \bar{x}_c = (x_c)_{init} \\ \bar{y}_c = (y_c)_{init} \\ \bar{\alpha}_h = \dfrac{1}{N}, \qquad h = 1, \ldots, N \\ \begin{pmatrix} \bar{r}_x \\ \bar{r}_y \end{pmatrix} = \bar{\mu} \begin{pmatrix} (r_x)_{init} \\ (r_y)_{init} \end{pmatrix}, \end{cases}$$

being

$$\bar{\mu} = \min_{h=1,\ldots,N} \sqrt{\frac{u_h((x_c)_{init}, (y_c)_{init})^2}{[(r_x)_{init}]^2} + \frac{v_h((x_c)_{init}, (y_c)_{init})^2}{[(r_y)_{init}]^2}} \tag{22}$$

and $\gamma, \mathcal{M}, \xi$ with proper positive values that guarantee the initial feasibility of (11)–(21), i.e. $\gamma \leq \bar{\mu}$ and $\max\{\bar{\mu}(r_x)_{init}\ ,\ \bar{\mu}(r_y)_{init}\} \leq \mathcal{M}/\xi$. After scaling both semiaxes by $\bar{\mu}$, every sampled contour point lies on or outside the initial ellipse. Equation (22) is the closed-form solution of a one-dimensional common-scaling problem, and requires only $O(N)$ arithmetic operations. If a purely conservative shrink is desired, one may replace $\bar{\mu}$ by $\min\{1, \bar{\mu}\}$. Note that broadly speaking the semiaxes $\bar{r}_x$ and $\bar{r}_y$ correspond to the semiaxes of the dashed–dotted ellipse in Figure 2, whose center coincides with the point $((x_c)_{init}, (y_c)_{init})$. Then, the solution of the nonlinear model in (11)-(21) yields a refinement of the four parameters r_x, r_y, x_c, and y_c that identify the required best fitting ellipse within the balloon. We also remark that the computation of $\bar{\mu}$ does not require solving any optimization problem, and reduces to $O(N)$ algebraic operations, being typically $N \in [60, 100]$.

We now specify in the model (11)–(21) the role played by each constraint/variable, and comment on the main implementation issues. In this regard, (see Figure 7) we highlight that $\mathcal{M}$ corresponds to the length (expressed in *pixels*) of the diagonal of the bounding box of the balloon (*i.e.*, the smallest rectangle in which Photoshop inscribes the balloon after scanning) plus a small displacement ε to allow a tolerance margin (in our setting $\varepsilon = 2$). The role of $\mathcal{M}$ and ξ is better clarified in the subsequent items. Hereafter, we comment on each of the nonlinear model (11)–(21) elements, giving the main motivations behind them.

- The objective function to maximize in (11) corresponds to having a large area for the text-safe ellipse, being indeed $\pi r_x r_y$ such area. This task is mainly pursued by introducing the constraints (17) (which are nonconvex). Basically, these last constraints impose that each point (P_{hx}, P_{hy}) must not lie within the final ellipse. The objective therefore enlarges the ellipse until one or more constraints become active; because the model is nonconvex, the result is a local maximizer in the restricted feasible region.
- The constraints (12)-(15) simply impose that the coordinates of the ellipse center lie within the convex hull of the sample points describing the contour of the balloon[3]. These constraints turn out to be essential in case the sample points on the boundary are almost collinear, and some ill-conditioning may arise. This situation can arise for narrow or nearly degenerate sampled contours. Note that these constraints are linear, which, in principle, makes them tractable. In any case, the relatively small number of unknowns in (11)–(21) (say $N + 4$) makes the overall solution process of the nonlinear model affordable, even when thousands of balloons are examined for each comics issue.
- The constraints (16) prevent the computation of undesirable local feasible solutions, and represent a tricky choice in the model (11)–(21). Indeed, observing that $\pi r_x r_y$ represents the area within the ellipse, and since the model is nonconvex, some local solutions which

[3] We remark that though the choice (12)-(15) appears a natural one, formally it does not ensure that in case of very irregular shapes of the balloon the final center of the ellipse will always lie inside its boundary.

are *undesirable* are expected. The bound constraints (16) aim at cutting a portion of the last local solutions. Figure 3 and Figure 4 motivate our choice: in the first figure, the initial ellipse (dashed contour of the shaded area) is endowed with the center $((x_c)_{init}, (y_c)_{init})$. Observe that some of the sample points on the boundary of the balloon are possibly inside the initial ellipse (shaded green ellipse with dashed contour). Conversely, the final feasible ellipse (shaded light blue ellipse with continuous red contour) found by the solver shows that the solver is indeed trapped at a local solution, which is definitely undesirable.
By contrast, in Figure 4 we can compare the initial ellipse (with dashed line contour and with center $((x_c)_{init}, (y_c)_{init})$) and the final desirable feasible ellipse (continuous line) found by the solver. We emphasize that, though the rotation angles (with respect to the abscissa axis of the bounding box) associated with the ellipses in Figure 4 coincide, the coordinates of their centers, along with the semiaxes, are possibly different, yielding remarkable degrees of freedom to the solver in order to improve the initial ellipse.
As a further consideration, the drawback detailed in Figure 3 can be partially alleviated through oversampling the points of the balloon contour (*i.e.*, increasing N). However, this choice increases solution time, since the number of constraints and unknowns of the nonlinear model grows linearly with N. Furthermore, we recall once more that the shape of the balloon might be unpredictable and cannot be altered when lettering.

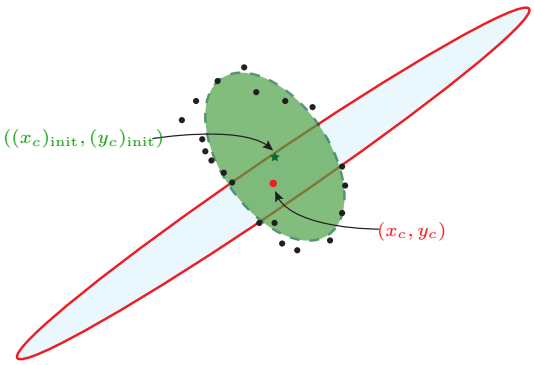


Fig. 3: The *initial* ellipse identified by the covariance matrix (shaded green ellipse with dashed contour) and the *final* feasible ellipse (with red continuous contour) corresponding to an undesirable local maximum, obtained by solving the nonconvex model (11)-(21). Observe that the points on the contour of the final balloon are all outside the shaded light blue ellipse, so that it indeed corresponds to a feasible solution of (11)-(21).

- Finally, the constraints (18)-(21) represent bound constraints on r_x, r_y, x_c, and y_c, and the rationale behind their introduction is that of again possibly cutting undesirable local solutions, as depicted in Figure 3. Observe that in our numerical experiments we set $\xi = 2$ and $\gamma = 0.1$, meaning that a displacement of 10% from the diagonal value $\mathcal{M}$ is tolerated. The value of γ was selected empirically from experiments on a substantial set of real balloons. Conversely, the choice of ξ has a more technical motivation, which is suggested in Figure 7. Here, two ellipses computed using the covariance matrix are shown, corresponding to two different balloons. The two ellipses show the same bounding box (thin continuous line rectangles). On the left side of the figure, the axes of the ellipse are definitely bounded by the lengths of the rectangle sides. By contrast, on the right side of the figure, the semiaxes are (almost) exactly bounded by $\mathcal{M}/2$.

Note that the bound constraints in the model (11)-(21) tend to reduce the area of the feasible region, so that we may deliberately cut off feasible solutions. However, this does not actually represent a practical drawback, since we noticed that this apparently sub-optimal assessment of the ellipse area does not affect the overall lettering procedure of the balloons

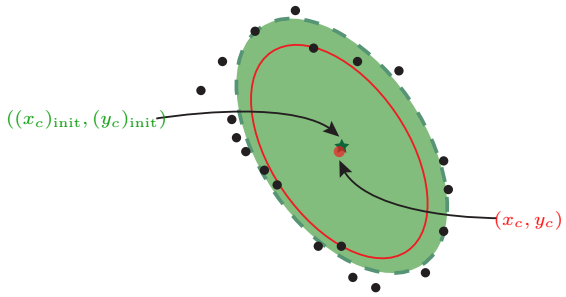


Fig. 4: The *initial* ellipse identified by the covariance matrix (dashed contour of the shaded area) and the feasible *final* ellipse corresponding to a desirable local maximum found by solving the nonconvex model (11)-(21) (continuous red contour). The points on the contour of the final balloon are all outside the found ellipse, so that the ellipse indeed corresponds to a feasible solution of (11)-(21). The centers of the two ellipses yet differ.

in the whole comic. Thus, the method deliberately trades a small amount of potentially usable area for faster and more stable computation.

4.4 Some remarks on solving the nonlinear model (11)–(21)

A local solution of the nonlinear model (11)–(21) is obtained iteratively applying the Sequential Least Squares Programming (SLSQP) solver (see *e.g.* [28]), adopting the powerful implementation provided by the `scipy.optimize` function in Python. SLSQP is appropriate for smooth constrained models of this size, although it provides local rather than global optimality guarantees. Its iterates also provide a useful visual record of how the ellipse changes during the search (examples are reported in Figure 11 and Figure 12, where the sequence of intermediate ellipses clearly indicates the maximization process).

It may be worth mentioning that the authors had initially considered an enhanced nonlinear model, where the rotation angle θ in (6) was among the unknowns to be assessed. However, the overall nonconvex model was definitely much more complex, so that the resulting number of local maximizers of the model was accordingly larger. This created several difficulties for the solver, so that multiple starting points (or a solver for global optimization) had to be considered in order to achieve a reasonably accurate final solution. On balance, this additional computational effort did not seem to justify the obtained improvement. Hence, we reverted to handling the simpler nonconvex model (11)-(21) that, within a reasonable amount of time, can be applied repeatedly across hundreds of balloons without requiring a global-optimization solve for every region.

As an additional issue, based on the contents in Section 1, we recall once more that both the shape and the layout of the balloons in comics are definitely associated with the artist/writer's creativity, meaning that balloons, as in Figure 5, can be difficult for any method based on a single convex text region. Highly irregular or composite balloons motivate future multi-region models. Similarly, an automatic procedure able to fully discriminate among *spurious balloons* is required, too. An example of such a case is provided in Figure 6, where the empty smaller balloons represent satellite bubbles forming the thought-balloon tail.

The finite bounds induced by ξ and γ make the searched region bounded. Feasibility is maintained by the explicitly constructed starting point whenever it also satisfies the area bounds in (16); these bounds should therefore be set consistently with the initial ellipse.

4.5 Text placement within the optimized ellipse

Here we deal with our second subproblem related to automatic balloon lettering, namely the text placement within the optimized text-safe ellipse inside each balloon. In this regard, we

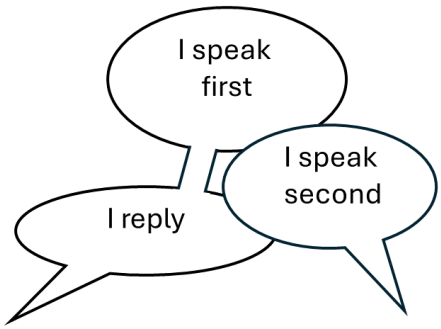


Fig. 5: Balloon shapes can be highly irregular in comics, causing difficulties to the automatic lettering procedures.

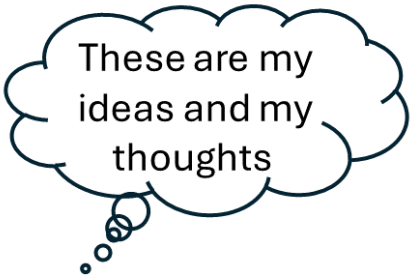


Fig. 6: The empty smaller balloons indicate *thoughts*, so that any automatic lettering procedure should be able to skip them and leave them blank.

separately focus on each balloon and repeatedly replicate the same procedure for all of them. In particular, the technique adopted to fill a balloon relies on the solution of an Integer Linear Programming (ILP) subproblem, which retains a number of appealing features suggesting its adoption. The optimization community well knows the following general aspects in favor of privileging an ILP model (see *e.g.* [30]–[32]) with respect to a nonlinear one:

- the linear-programming relaxation used inside branch-and-bound or branch-and-cut is convex, so it provides globally valid bounds for the discrete search;
- MILP solvers are mature, robust, and computationally efficient;
- solvers for nonlinearly constrained optimization problems may require the user to provide a feasible starting point (see also Section 6.2), whose assessment might be far from being trivial. Conversely, solvers for ILP subproblems implement iterative procedures that do not require any initialization from the user, so that even non-specialist practitioners can use them without any specific insight into their technicalities.

Now, correctly identifying the ILP subproblem for lettering the k-th balloon requires some combined information, encompassing the next few specific elements:

1. the exact text sentence (say the entire string of characters including blank spaces and hyphenation) to be embedded within the balloon, as coming from either an expert translator or an automatic procedure (see also Section 1 for details on this task);
2. the correct hyphenation of the text, as from an automatic tool applied to the current target language of choice;

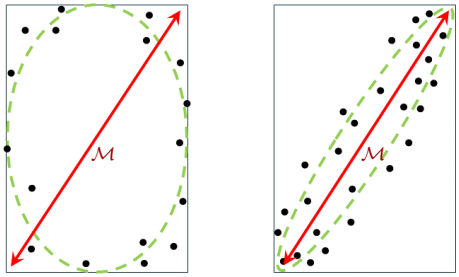


Fig. 7: The bounding boxes (the identical rectangles) are associated with two different ellipses, corresponding to different balloons. The quantity $\mathcal{M}$ represents the diagonal of the bounding boxes plus a small displacement ε. In both cases, the semiaxes of the ellipse are smaller than $\mathcal{M}/2$.

3. the font F (*i.e.*, the dimension) of the used characters, as expressed in *typographic points* (say *pixels*);
4. the length (*i.e.*, the number of pixels) of each syllable;
5. the overall number $m \in \mathbb{N}$ of syllables giving the text sentence;
6. the sequence $\{\ell_1^F, \ldots, \ell_m^F\}$ (*i.e.*, the vector ℓ^F for short) of the m lengths (expressed in *pixels*) associated to the m syllables and the font size F;
7. the maximum number n_F of lines allowed to fit the text sentence within the k-th balloon;
8. given the k-th balloon and the number n_F of lines, we consider the height b_k of the bounding box of the k-th balloon, along with the vector a_k, whose entries $\{(a_k)_j\}$ are computed as follows. Dividing the maximal area ellipse inside the k-th balloon (obtained by solving the nonlinear model (11)–(21)) in n_F lines, the entry $(a_k)_j$ represents the length (expressed in *pixels*) of the j-th line, for any $j = 1, \ldots, n_F$[4];
9. given the k-th balloon, we consider the vector w_k with exactly n_F entries, such that

   if n_F is *odd* then
   $$w_k = \left(\frac{n_F+1}{2}, \ldots, 2, 1, 2, \ldots, \frac{n_F+1}{2}\right)^T;$$

   if n_F is *even* then
   $$w_k = \left(\frac{n_F}{2}, \ldots, 2, 1, 1, 2, \ldots, \frac{n_F}{2}\right)^T;$$

10. given the k-th balloon and the exact text sentence (*i.e.*, the string of m syllables including blank spaces and hyphenation) to be embedded within the balloon, we define the two m-dimensional vectors h_k and s_k (their role will be better clarified later on), such that for any $i = 1, \ldots, m$:

    $(h_k)_i = \Delta^F$ if the i-th syllable is followed by *another syllable*,

    $(h_k)_i = 0$ if the i-th syllable is followed by the *blank space*,

    $(s_k)_i = 0$ if the i-th syllable is followed by *another syllable*,

    $(s_k)_i = \Gamma^F$ if the i-th syllable is followed by the *blank space*.

[4]We have been fast when detailing a_k and b_k, since the procedure yielding them, to large extent, represents an implementation issue and does not involve any optimization procedure.

Note that $(h_k)_m = (s_k)_m = 0$, for any k, Δ^F is the width of the *hyphenation character*, and Γ^F is the width of the *blank space character* (both Δ^F and Γ^F are associated with the font size F). Observe that not all the above quantities can be independently chosen. Indeed, since F represents the size of the adopted character font, then the values of b_k and n_F must satisfy the necessary geometric condition

$$n_F \leq \left\lfloor \frac{b_k + \delta}{F + \delta} \right\rfloor, \tag{23}$$

being δ the line spacing distance (more precisely, observe that the value b_k in (23) should be replaced by the height of the ellipse found by solving (11)-(21)). Given the above elements, the next model details our ILP subproblem adopted to predict the correct fitting of any syllable within each one of the text lines in the k-th balloon, so that standard rules for sequential fluent reading are preserved. In this regard, we first define the set of variables ($i = 1, \ldots, m$, $j = 1, \ldots, n_F$):

$$y_{i,j} = \begin{cases} 1 & \text{if the } i\text{-th syllable is located in the } j\text{-th line,} \\ \\ 0 & \text{otherwise,} \end{cases}$$

$$z_{i,j} = \begin{cases} 1 & \text{if the } i\text{-th syllable is located in the } (j-1)\text{-th text line} \\ & \text{and the } (i+1)\text{-th syllable is located in the } j\text{-th text line,} \\ \\ 0 & \text{otherwise.} \end{cases}$$

Note that the variables $\{z_{i,j}\}$ have the purpose to account for the correct segmentation of the text sentence on different lines, within the balloon.

Now, let us introduce the constant value $Big_M \gg 1$ (big M). Define the set of central candidate rows

$$\mathcal{J}_c = \begin{cases} \{(n_F + 1)/2\}, & n_F \text{ odd,} \\ \\ \{n_F/2, n_F/2 + 1\}, & n_F \text{ even.} \end{cases} \tag{24}$$

For notational convenience set $z_{i,1} = 0$ and $z_{i,n_F+1} = 0$; no line transition can enter the first row or leave the last row. Then, given the values of F, m, and n_F (we assume for the sake of brevity $n_F > 1$, since one-line balloons -*e.g.* those balloons containing exclamations- are easily separately treated), we propose the next ILP model to fill the largest blank area by the text, within the k-th balloon.

$$\min_{y,z} \sum_{i=1}^{m} \sum_{j=1}^{n_F} \left(y_{i,j} + \frac{(w_k)_j}{i+1} z_{i,j} \right) + \sum_{j=1}^{\lfloor n_F/2 \rfloor} [n_F - (w_k)_j] [L_j - L_{j+1}] + \sum_{j=\lfloor n_F/2 \rfloor + 1}^{n_F - 1} [n_F - (w_k)_j] [L_{j+1} - L_j], \tag{25}$$

$$y_{i+1,j} \leq y_{i,j} + z_{i,j} Big_M, \qquad i = 1, \ldots, m-1, \quad j = 1, \ldots, n_F, \tag{26}$$

$$y_{i+1,j} \leq y_{i,j-1} + (1 - z_{i,j}) Big_M, \qquad i = 1, \ldots, m-1, \quad j = 2, \ldots, n_F, \tag{27}$$

$$z_{i,j} \leq y_{i,j-1}, \qquad i = 1. \ldots, m-1, \quad j = 2, \ldots, n_F, \tag{28}$$

$$z_{i,j} \leq y_{i+1,j}, \qquad i = 1. \ldots, m-1, \quad j = 2, \ldots, n_F, \tag{29}$$

$$z_{i,j} \geq y_{i,j-1} + y_{i+1,j} - 1, \qquad i = 1\ldots, m-1, \quad j = 2, \ldots, n_F, \tag{30}$$

$$\sum_{j \in \mathcal{J}_c} y_{\lceil m/2 \rceil, j} = 1, \tag{31}$$

$$\sum_{i=1}^{m} \left[\ell_i^F + (s_k)_i\right] y_{i,j} + \sum_{i=1}^{m} \left[(h_k)_i - (s_k)_i\right] z_{i,j+1} \leq (a_k)_j,$$

$$j = 1, \ldots, n_F, \tag{32}$$

$$\sum_{j=1}^{n_F} y_{i,j} = 1, \qquad i = 1, \ldots, m, \tag{33}$$

$$y_{i,j} \in \{0,1\}, \quad z_{i,j} \in \{0,1\}, \quad z_{m,j} = 0, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n_F, \tag{34}$$

where L_j represents the length of the *text* in the j-th line, *i.e.*,

$$L_j = \sum_{i=1}^{m} \ell_i^F y_{i,j}.$$

The model (25)–(34) combines three layout goals. The objective was the result of an empirical analysis. It penalizes line breaks, especially when they occur far from the central row(s), and its line-length terms favor the familiar balloon profile in which line lengths grow toward the middle and then decrease. The disjunctive constraints (26)–(27) use Big_M and $z_{i,j}$ to preserve the sequential order of syllables while allowing a transition only between adjacent rows; furthermore, (28)–(30) force a nonzero value of the variable $z_{i,j}$ in case consecutive syllables in a word change line. Relation (31) anchors the central syllable to a central candidate row. A less restrictive implementation on the central syllable could possibly be considered; however, in our numerical experience (31) proved to be a good compromise. Capacity constraints (32) account for glyph widths, blank spaces, and end-of-line hyphenation, whereas (33)–(34) assign every syllable to exactly one row. Thus the optimization does not merely ask whether the sentence fits: among feasible layouts, it favors a compact, visually balanced arrangement. Observe that by (33) the objective function (25) can be simplified eliminating the explicit dependency on $\{y_{i,j}\}$ in the first sum, though this arrangement does not seem to bring an appreciable computational gain.

In the reported experiments we used $Big_M = 10^6$. The small instances were empirically insensitive to this value, although implementation-quality models should scale Big_M as tightly as possible. A weaker aggregate assignment equation, $\sum_{i=1}^{m} \sum_{j=1}^{n_F} y_{i,j} = m$, was also tested, but the computational saving was negligible for the short sentences encountered in the balloons; the explicit per-syllable equations were therefore retained for clarity and robustness.

5 Modeling clarifications, derivations, and worked examples

This section collects a few derivations that are useful for implementation and for interpreting the optimization output.

5.1 PCA orientation, consistency of rotated coordinates and scaling

The covariance matrix in (4) is computed in pixel units. As an example, if the principal eigenvector is $V_1 = (0.866, 0.5)^T$, then (6) gives $\theta \simeq 30°$. A contour point cannot then be tested against the ellipse using its original horizontal and vertical offsets. It must first be rotated through (8)–(9). For example, if $p_h - (x_c, y_c)^T = (20, 10)^T$, then the aligned

coordinates are approximately $u_h = 22.32$ and $v_h = -1.34$. For semiaxes $r_x = 25$ and $r_y = 12$, the left-hand side of (10) is about 0.81, so this sampled contour point would lie inside the candidate ellipse and the candidate would be infeasible. The example shows why the rotation estimated by PCA must be propagated consistently into every containment constraint.

Let now $(r_x)_{init}, (r_y)_{init}$ be the PCA initial semiaxes estimates and keep the initial center fixed. A common scaling factor $\mu > 0$ gives semiaxes $\mu(r_x)_{init}$ and $\mu(r_y)_{init}$. For sampled point h, define

$$q_h = \frac{u_h^2}{(r_x)_{init}^2} + \frac{v_h^2}{(r_y)_{init}^2}. \tag{35}$$

The scaled ellipse excludes all sampled contour points if $q_h/\mu^2 \geq 1$ for every h. Hence the largest common scale satisfying these inequalities is

$$\bar{\mu} = \min_{h=1,\dots,N} \sqrt{q_h}. \tag{36}$$

Using $\min\{1, \bar{\mu}\}$ gives a purely conservative shrink. This computation requires $O(N)$ arithmetic operations and supplies a feasible starting ellipse whenever the remaining area and bound constraints are consistent with it.

5.2 What the nonlinear model certifies

From an optimization perspective, ellipsoidal geometry and local nonlinear models connect this construction to established optimization treatments [44, 45]. The inequalities in (10) are enforced only at the sampled contour points. Therefore a feasible solution means that no *sampled* boundary point lies inside the ellipse. It does not prove continuous containment between samples, and SLSQP does not prove global optimality for the nonconvex formulation. These distinctions matter most for sharp concavities or sparse sampling. Increasing N reduces geometric discretization risk but increases the number of nonlinear constraints approximately linearly. A practical implementation can therefore use adaptive sampling: dense points near high-curvature portions of the contour and fewer points on nearly straight segments.

5.3 Interpretation of the integer formulation (25)-(34)

The binary variable $y_{i,j}$ assigns syllable i to row j, while $z_{i,j}$ marks a transition between consecutive rows. The assignment equations ensure that each syllable appears exactly once; the sequencing constraints prevent the text from moving backward or skipping rows; and (32) limits the occupied width by the row capacity. The objective favors line breaks near the vertical center and line lengths that grow toward the center and shrink afterwards, reproducing a common balloon-lettering profile. The formulation is a small ILP, for which branch-and-bound/branch-and-cut technology is mature [30–32].

A compact alternative is to introduce the row index $q_i = \sum_j j y_{i,j}$ and impose

$$0 \leq q_{i+1} - q_i \leq 1, \tag{37}$$

together with $\sum_j y_{i,j} = 1$. Equation (37) states directly that the next syllable remains on the current line or advances by one line. Transition binaries are still useful when a hyphenation or line-break penalty must be charged, but the compact view clarifies the combinatorial structure and can reduce dependence on very large Big_M values.

5.4 Worked font-size search and numerical scaling

Suppose the initial candidate font is 20 pt. The rendering engine measures every syllable at that size, the row height determines admissible values of n_F, and the chord formula gives

row capacities. If the ILP is infeasible for both odd and even row counts, the algorithm tests 19 pt, then 18 pt, and so on. If 19 pt is infeasible and 18 pt is feasible, 18 pt is the largest feasible size encountered by the monotone font size search. This outer loop is inexpensive in the reported experiments because the text-layout ILPs are cheaper relative to the nonlinear ellipse solve.

The source implementation used $Big_M = 10^6$ and was empirically insensitive to substantial reductions for its small instances. Nevertheless, unnecessarily large constants can weaken LP relaxations and amplify numerical error [32]. Bounds derived from actual row capacities are preferable when the formulation is transferred to larger text blocks. The same principle applies to the nonlinear stage: all distances are kept in pixel units, scale-dependent bounds use the balloon bounding-box diagonal, and stopping tolerances should be reported together with image resolution.

6 Software and implementation issues

The prototype `OptiComics` is designed to insert translated text into editable comic pages rather than to flatten a generated result directly into the artwork. This choice supports proofreading: a translator or editor can revise wording after the first typesetting pass without reconstructing the entire image. The reference workflow uses Adobe Photoshop layers and its scripting interface [46]; the optimization modules themselves are independent of that graphics application.

The required inputs are (i) comic pages in a layered format, with the text regions prepared for lettering, and (ii) a plain-text file in which each line corresponds to the sentence associated with a balloon or text panel. An optional color-to-font map stores typographic presets. The preparation stage intentionally does not solve OCR, automatic text removal, or artwork inpainting; those are upstream tasks and can be supplied by other document-analysis methods.

6.1 Preparation and balloon masks

For each target region, the user or an upstream detector prepares a layer whose color identifies the balloon and, when desired, a font/style preset. The region is converted to grayscale and thresholded to a binary mask. OpenCV contour extraction then produces the boundary samples used by Section 4.2 [41, 47]. Excessively dense contours are subsampled because each retained point becomes a nonlinear constraint. In typical tests, N is on the order of a few dozen to about one hundred points.

6.2 Ellipse computation

The numerical implementation relies on the standard scientific Python array ecosystem, for which NumPy provides the underlying array representation [48]. The sampled contour is passed to the PCA routine, which returns the initial center, orientation, and scale. The common shrink in (36) produces a sample-consistent starting ellipse, after which the nonlinear model is solved through SLSQP. The implementation follows the standard sequential quadratic programming framework [28, 49]; the reported experiments use SciPy's optimization infrastructure [50].

A practical readability rule is applied to small rotations. If the PCA angle differs only slightly from horizontal, the text is kept horizontal rather than reproducing a minor tilt of the balloon. Experimental reading research indicates that rotated letters can increase reading difficulty [42, 43]. The rule sacrifices a small amount of geometric fit in exchange for more conventional typography.

The source experiments generally used an SLSQP tolerance of 10^{-3}, for which a balloon solve typically required less than one second. A small number of difficult cases required a much tighter tolerance, around 10^{-8}, and consequently longer runs. Exact timings depend on

image size, contour complexity, initialization, and software version; therefore solver tolerance and resolution should be recorded in reproducibility reports.

Algorithm 1: Procedure to find the approximate maximum inscribed ellipse.

Input : $\mathbb{B}$
Output: $x_c^*, y_c^*, r_x^*, r_y^*, \theta^*$

begin
  $\mathbb{I} \leftarrow \text{grayscale}(\mathbb{B})$;
  $s \leftarrow \text{approx_contour}(\mathbb{I})$;
  $(x_c)_{init}, (y_c)_{init} \leftarrow \text{centroid}(s)$;
  $\Sigma \leftarrow \text{Covariance}(s)$;
  $\lambda \leftarrow \text{eigenvalues}(\Sigma)$;
  if $\Sigma_{1,2} = 0$ ***and*** $\Sigma_{1,1} \geq \Sigma_{2,2}$ **then**
    $\theta^* \leftarrow 0$;
  else if $\Sigma_{1,2} = 0$ ***and*** $\Sigma_{1,1} < \Sigma_{2,2}$ **then**
    $\theta^* \leftarrow \pi/2$;
  else
    $\theta^* \leftarrow \texttt{atan2}\left(\frac{\lambda_1 - \Sigma_{1,1}}{\Sigma_{1,2}}\right)$;
  end
  $(r_x)_{init} \leftarrow \sqrt{\lambda_1}$;
  $(r_y)_{init} \leftarrow \sqrt{\lambda_2}$;
  $x_c^*, y_c^*, r_x^*, r_y^* \leftarrow$ solve the max. problem (11)–(21);
end

6.3 Text placement and font search

After the ellipse has been computed (see Algorithm 1, with an intuitive explanation of the symbols), the target sentence is divided into hyphenation units. For a tentative font size F, the rendering engine measures the width of each unit, the blank-space character, and the hyphen. The ellipse is sliced into candidate rows and the ILP in Section 4.5 is then solved: (1) for an odd number of rows and, if needed, (2) for an even number. If neither case is feasible, F is reduced by one typographic point and the process repeats (see Algorithm 2, with an intuitive explanation of the symbols and the parameters).

The search begins from a large candidate font (60 pt in the reference implementation) and stops when a feasible placement is found or when the font would fall below 6 pt. At the 72-dpi resolution used in the source experiment, one typographic point corresponds to one pixel, which simplifies the conversion between text measurements and raster geometry. For other resolutions, all widths and heights must be converted consistently before model construction.

The source study reports CBC/COIN–OR as the mixed-integer solver used for the text-layout formulation. The mathematical model is solver independent. For reproducibility, the final software release should record the exact Python modeling interface, solver version, and parameter settings because interfaces and default back ends can change over time. In the chapter-scale experiment, the integer stage is substantially faster than the nonlinear ellipse stage.

6.4 Why editability is part of the implementation design

The last step writes the chosen text back to editable graphics layers and centers the resulting block near the optimized ellipse center. This is deliberately separated from the optimization itself. The mathematical model can propose line breaks and font size, while a final editorial

Algorithm 2: Procedure to place the text.

Input : $\mathbb{S}$
Output: P_{best}, F_{best}

begin
  $F \leftarrow \bar{F}$;
  $\mathbb{S} \leftarrow \{s_i \ : \ 1 \leq i \leq m\}$;
  $\mathrm{P}_{\text{best}} \leftarrow \emptyset$;
  $\mathrm{F}_{\text{best}} \leftarrow \emptyset$;
  while $P_{best} = \emptyset$ ***and*** $F \geq 6$ **do**
    $D \leftarrow \mathbb{S} \times F$;
    $\ell\ell^F \leftarrow$ calc_lines(F, r_x, r_y, θ, "odd");
    $\mathrm{P}_F \leftarrow$ solve$\big((25)$-$(34), m, n_F, \ell\ell^F, D\big)$;
    if $P_F \neq \emptyset$ **then**
      $\mathrm{P}_{\text{best}} \leftarrow \mathrm{P}_F$;
      $\mathrm{F}_{\text{best}} \leftarrow F$;
    else
      $\ell\ell^F \leftarrow$ calc_lines(F, r_x, r_y, θ, "even");
      $\mathrm{P}_F \leftarrow$ solve$\big((25)$-$(34), m, n_F, \ell\ell^F, D\big)$;
      if $P_F \neq \emptyset$ **then**
        $\mathrm{P}_{\text{best}} \leftarrow \mathrm{P}_F$;
        $\mathrm{F}_{\text{best}} \leftarrow F$;
      end
    end
    $F \leftarrow F - 1$;
  end
end

pass can correct translation, punctuation, style, or exceptional graphic effects. Keeping these layers editable also makes the tool useful in production settings where changes arrive late in the workflow. In operational terms, optimization acts as a repeatable decision-support component rather than an irreversible image-generation step.

7 Results

We evaluated `OptiComics` on complete and partial production workflows involving both East Asian and European source material translated into English. The purpose of these experiments is twofold. First, they show whether the geometric and discrete optimization stages can be embedded in a realistic lettering process rather than tested only on synthetic shapes. Second, they expose the practical bottlenecks of the pipeline, which are important from an operations-research perspective because the optimization models are only one component of the total production time.

7.1 Qualitative page-level examples

Figure 8 illustrates the processing of page 3 from chapter 1214 of the Japanese manga *Hajime no Ippo*. The source page follows the usual manga navigation convention, with the page read predominantly from top to bottom and right to left. Panel (a) is the original page. In the preparation stage, panel (b), the text areas are cleaned and the balloon masks are encoded so that the appropriate typographic style can be retrieved. The optimization modules then compute a sample-consistent text-safe ellipse for every selected balloon and solve the line-assignment model at the largest feasible font size. Panel (c) exposes the resulting text placement, whereas panel (d) shows the final clean-up and centering step. In routine

execution, the geometric construction is internal and the user does not need to inspect the ellipses explicitly.

Figure 9 repeats the experiment on page 5 of chapter 105 of the Italian comic *Rat-Man*. This example is deliberately different from the Japanese one: the source is read left to right, and the balloons have a noticeably different geometric distribution. The same optimization architecture can nevertheless be used because reading order is represented in the prepared layer sequence, whereas each balloon is treated geometrically in its own local coordinate system. This separation is one of the practical advantages of the formulation: the continuous and integer models do not require a language-specific redesign when the source writing direction changes.

A more demanding balloon geometry is shown in Figure 10, using a page from the Japanese manga *Letter Bee*. Several dialogue balloons are elongated and provide relatively narrow horizontal chords. The available row capacities therefore vary strongly with vertical position. The example demonstrates why a simple rectangular text box can be inadequate: a line that fits near the center of the balloon may exceed the admissible width close to its upper or lower boundary. The row-capacity construction in Section 4.5 explicitly accounts for this variation.

The two optimization sequences in Figures 11 and 12 provide a visual interpretation of the nonlinear stage on two challenging balloons from *Gachiakuta*. The intermediate ellipses make clear that SLSQP is not solving a closed-form inscribed-ellipse problem: it iteratively adjusts the center and semiaxes subject to the sample-based boundary constraints and the additional bounds introduced to exclude undesirable local solutions. These sequences should therefore be read as solver trajectories for the adopted nonconvex model, not as certificates of global optimality.

7.2 Full-chapter timing experiment

The full workflow was also executed on chapter 1214 of *Hajime no Ippo*. The experimental set contains 15 nonblank pages and 162 balloons, corresponding to 10.8 ± 5.07 balloons per page. The English text contains on average 9.85 ± 7.13 syllables per balloon, with median 8.5. Computations were performed on a machine equipped with an AMD Ryzen 7 3700X CPU, 64 GB of RAM, and an NVIDIA RTX 3070 GPU. Table 1 reports the wall-clock decomposition recorded in the source implementation.

The complete chapter required 346.757 s, i.e., 23.117 s per page on average. Of this total, 172.449 s (49.73%) were spent in optimization and 173.085 s (49.92%) in the Photoshop placement interface; preparation accounted for only 1.223 s. Within the optimization time, ellipse computation required 107.980 s and text-layout optimization 64.469 s. Thus the geometric stage used about 1.675 times the time of the integer-layout stage, or approximately 67.5% more. This confirms the practical observation that the nonconvex geometric subproblem, rather than the ILP, is the dominant mathematical-programming cost in the tested implementation.

A simple linear extrapolation of the measured per-page time gives approximately 69.4 minutes for 180 pages and 77.1 minutes for 200 pages. Such an extrapolation is only indicative: page complexity, number of balloons, font measurement, and interaction with the graphics application vary across a volume. It is nevertheless useful for identifying the scale of the potential production benefit. It also shows why optimizing only the mathematical subproblems is insufficient: because almost half of the observed time is spent in the application interface, improvements to the graphics-software communication layer could be as important as improvements to the nonlinear solver.



(a) Starting page: the original comics page



(b) Prepared page: balloons are detected, and different colors identify different font types for the text

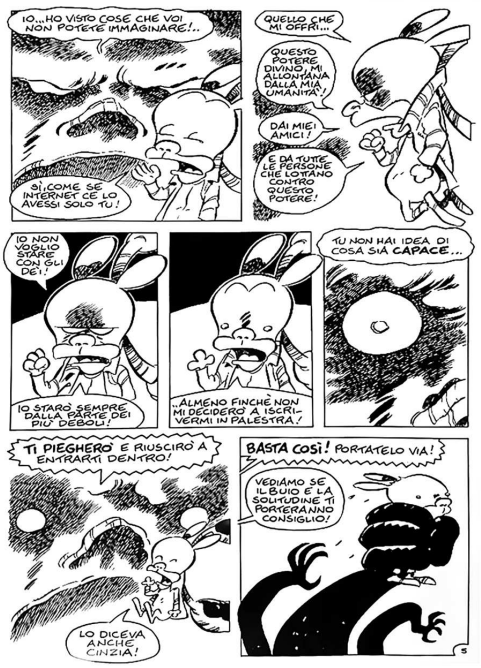


(c) Text placement in the balloons

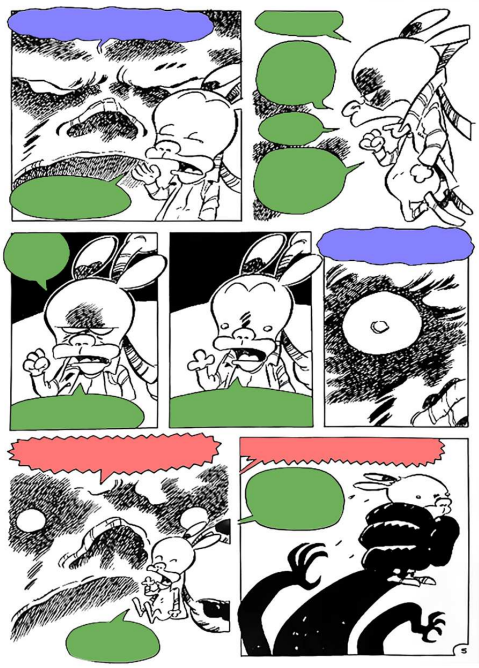


(d) Clean up and centering: the entire text layout in each balloon is possibly slightly moved, to compensate for irregularities of the balloon boundary

Fig. 8: Demo of the placement steps for a real-world Japanese manga's chapter page (*original language writing top-to-bottom*). Source: Hajime no Ippo, chapter 1214, page 3 (Published by Kōdansha, 2018).



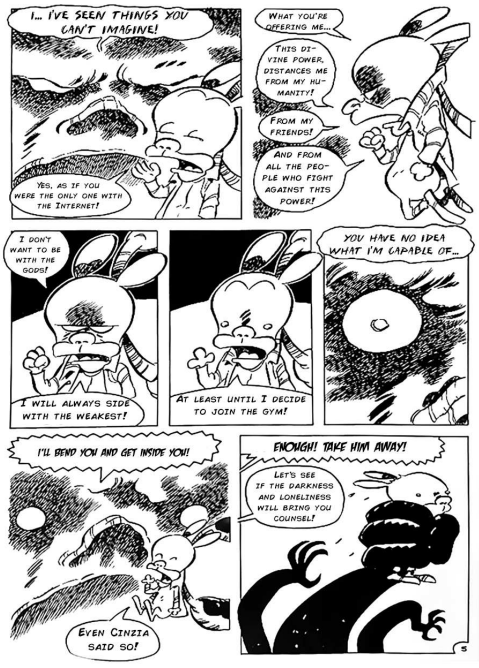
(a) Starting page: the original comics page



(b) Prepared page: balloons are detected, and different colors identify different font types for the text



(c) Text placement in the balloons



(d) Clean up and centering: the entire text layout in each balloon is possibly slightly moved, to compensate for irregularities of the balloon boundary

Fig. 9: Demo of the placement steps for a real-world Italian comic's chapter page (*original language writing left-to-right*). Source: Rat-Man, chapter 105, page 5 (Published by Panini Comics, 2015).

Stage	Total Time (s)	% of Total	Page avg (s)
Preparation	1.223	0.35%	0.082
Optimizations	172.449	49.73%	11.497
├ Ellipses opt.	107.980	31.14%	7.199
└ Text opt.	64.469	18.59%	4.298
Photoshop Placement	173.085	49.92%	11.539
Total	**346.757**	**100.00%**	**23.117**

Table 1: Execution times for `OptiComics` on a full real-world chapter, Hajime no Ippo chapter 1214, totaling 15 pages and 162 balloons, with an average of 10.8 ± 5.07 balloons per page.



(a) Starting page: the original comics page

(b) Clean up and centering: the entire text layout in each balloon is possibly slightly moved, to compensate for irregularities of the balloon boundary

Fig. 10: Demo for lettering a real-world Japanese manga's chapter page (*original language writing top-to-bottom*). Source: Letter Bee, chapter 56, page 7 (Published by Shūeisha, 2012).

8 Conclusions and future work

We presented `OptiComics`, an optimization-based workflow for lettering translated comics while preserving the geometry of the original page. The method separates a continuous geometric decision from a discrete typographic decision. PCA supplies an orientation and scale estimate; a nonconvex nonlinear program refines a large sample-consistent text-safe ellipse; and an ILP allocates ordered syllables to rows while accounting for spaces, hyphenation,

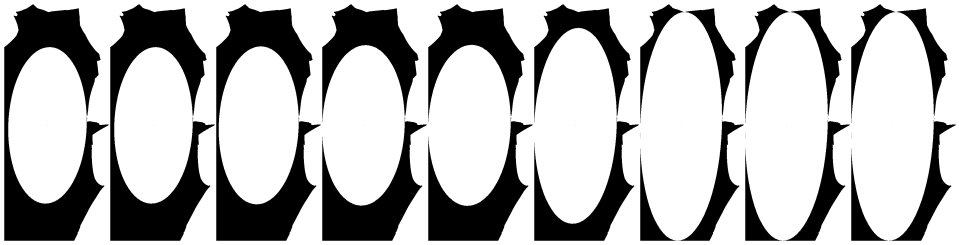


Fig. 11: Gachiakuta chapter 2 page 19 balloon 1. Ellipse optimization in 24 steps.

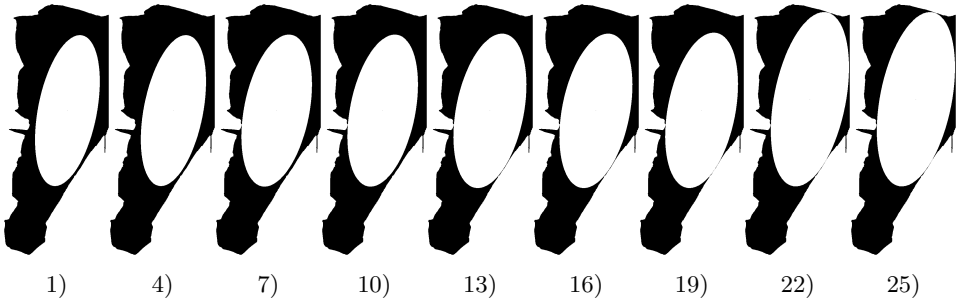


Fig. 12: Gachiakuta chapter 23 page 15 balloon 2. Ellipse optimization in 25 steps.

row capacities, central placement, and line balance. The formulations are intentionally interpretable and can be inserted between independent modules for segmentation/translation and final graphics placement.

The experiments show that this decomposition is practical at chapter scale. For the 15-page, 162-balloon *Hajime no Ippo* experiment, the complete workflow takes 346.757 seconds on the reference computer. About half of this time is spent in optimization and half in interaction with the graphics application, indicating that production efficiency depends as much on software integration as on solver speed. The examples from *Rat-Man*, *Letter Bee*, and *Gachiakuta* further show that the same core models can handle different reading directions and markedly different balloon shapes.

Several limitations are explicit. The nonlinear problem is nonconvex, SLSQP provides a local solution, and containment is checked on sampled contour points rather than the exact continuous boundary. A single ellipse can also be conservative for highly irregular or composite balloons. A natural extension is therefore to cover the useful region with several rectangles, ellipses, or other convex components and to coordinate text allocation across them. Adaptive contour sampling and global or multistart optimization are further options for difficult cases. If the target language itself requires vertical lettering, the row-capacity and ordering components should be generalized to vertical columns rather than merely rotating a Western layout.

Machine-learning methods can improve upstream balloon recognition and exceptional-case classification, while learned quality models could help predict font or line-break preferences. Such additions should complement rather than obscure hard geometric constraints. The current workflow also deliberately leaves linguistic and cultural decisions to translators and editors. Future work can connect the optimization layer to tools for onomatopoeia, culturally specific text, speaker association, and semantic consistency, drawing on translation and comics research [7–9, 13, 14, 51].

As a qualitative comparison, the source study also tested general-purpose generative workflows on the Japanese page used in Figure 8. The outputs in Figure 13 altered aspects of page content or produced unacceptable translation/lettering, despite an explicit preservation



(a) Exploratory output generated with ChatGPT 5.1 [52]

(b) Exploratory output generated with ChatGPT 5.2 [52]

Fig. 13: Exploratory general-purpose generative outputs obtained from the source page used in Figure 8. The prompt requested English translation while preserving page layout, pictures, balloon position, and balloon size. The examples illustrate the distinction between prompt-level preservation and the hard geometric constraints imposed by `OptiComics`.

prompt. This experiment is not a benchmark of generative AI and should not be read as a general performance claim; model versions and capabilities evolve rapidly. It nevertheless motivates the methodological choice adopted here: when the editorial requirement is that a balloon contour or drawing must not move, encoding that requirement as a mathematical constraint is fundamentally different from requesting it through natural-language prompting.

Future evaluation should include professional letterers and readers. Eye-movement and comprehension studies could assess whether automatically generated layouts differ perceptually from professional ones [43, 53, 54], while production studies could measure the frequency and type of human overrides. Such evidence would complement the current feasibility and timing results, and help convert the present optimization framework into a more complete, human-centered publishing tool.

9 Statements and Declarations

Competing Interests. Alessandro Bicciato declares no relevant financial or non-financial interests. Giovanni Fasano is an Associate Editor of Operations Research Forum and was not involved in the editorial handling of this manuscript.

Funding. The authors received no financial support for the research, authorship, or publication of this article.

Author contributions. Both authors contributed to the conception and mathematical formulation of the study, the interpretation of the computational results, and the writing and revision of the manuscript. The software implementation and computational workflow were developed within the joint project described in the paper.

Data and materials availability. The aggregate computational results are reported in the manuscript. The visual examples and figure files used for the experiments are included with the submission source files.

Code availability. The algorithms and implementation architecture are described in sufficient detail to reproduce the mathematical workflow. The source implementation can be made available by the authors on reasonable request.

References

[1] Cohn, N.: The Visual Language of Comics: Introduction to the Structure and Cognition of Sequential Images. Bloomsbury Academic, London (2013)

[2] Brienza, C.: Producing comics culture: a sociological approach to the study of comics. Journal of Graphic Novels and Comics **1**(2), 105–119 (2010) https://doi.org/10.1080/21504857.2010.528638

[3] Oittinen, R.: Translating for Children. Garland Publishing, New York (2000). https://doi.org/10.4324/9780203902004

[4] Fischer, B., Jensen, M.N. (eds.): Translation and the Reconfiguration of Power Relations: Revisiting Role and Context of Translation and Interpreting vol. 7. LIT Verlag, Berlin (2012)

[5] McCloud, S.: Making Comics: Storytelling Secrets of Comics, Manga and Graphic Novels. HarperCollins Publishers, London (2006). https://books.google.it/books?id=uWQ6PwAACAAJ

[6] Mikkonen, K.: The Narratology of Comic Art. Routledge, New York (2017)

[7] Kaindl, K.: Thump, whizz, poom: A framework for the study of comics under translation. Target **11**(2), 263–288 (1999) https://doi.org/10.1075/target.11.2.05kai

[8] Celotti, N.: The translator of comics as a semiotic investigator. In: Zanettin, F. (ed.) Comics in Translation, pp. 33–49. St. Jerome Publishing, Manchester (2008)

[9] Borodo, M.: Multimodality, translation and comics. Perspectives **23**(1), 22–41 (2015) https://doi.org/10.1080/0907676X.2013.876057

[10] Zanettin, F. (ed.): Comics in Translation. St. Jerome Publishing, Manchester (2008)

[11] Bryan, P.C.: Creation, Translation, and Adaptation in Donald Duck Comics: The Dream of Three Lifetimes. Palgrave Macmillan, Cham (2021). https://doi.org/10.1007/978-3-030-73636-1

[12] Epstein, B.J.: Translating Expressive Language in Children's Literature: Problems and Solutions. Peter Lang, Oxford (2012). https://doi.org/10.3726/978-3-0353-0271-4

[13] Guynes, S.A.: Four-color sound: A Peircean semiotics of comic book onomatopoeia. Public Journal of Semiotics **6**(1), 58–72 (2014)

[14] Pratha, N.K., Avunjian, N., Cohn, N.: Pow, punch, pika, and chu: The structure of sound effects in genres of american comics and japanese manga. Multimodal Communication **5**(2), 93–109 (2016) https://doi.org/10.1515/mc-2016-0017

[15] Augereau, O., Iwata, M., Kise, K.: An overview of comics research in computer science. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 3, pp. 54–59 (2017). IEEE

[16] Rigaud, C., Tsopze, N., Burie, J.-C., Ogier, J.-M.: Robust frame and text extraction from comic books. In: Graphics Recognition. New Trends and Challenges: 9th International Workshop, GREC 2011, Seoul, Korea, September 15-16, 2011, Revised Selected Papers, pp. 129–138 (2013). https://doi.org/10.1007/978-3-642-36824-0_13 . Springer

[17] Rigaud, C., Burie, J.-C., Ogier, J.-M., Karatzas, D., Van de Weijer, J.: An active contour model for speech balloon detection in comics. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1240–1244 (2013). https://doi.org/10.1109/ICDAR.2013.251 . IEEE

[18] Rigaud, C., Burie, J.-C., Ogier, J.-M.: Text-independent speech balloon segmentation for comics and manga. In: Graphic Recognition. Current Trends and Challenges: 11th International Workshop, GREC 2015, Nancy, France, August 22–23, 2015, Revised Selected Papers 11, pp. 133–147 (2017). https://doi.org/10.1007/978-3-319-52159-6_10 . Springer

[19] Guérin, C., Rigaud, C., Mercier, A., Ammar-Boudjelal, F., Bertet, K., Bouju, A., Burie, J.-C., Louis, G., Ogier, J.-M., Revel, A.: eBDtheque: a representative database of comics. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1145–1149 (2013). https://doi.org/10.1109/ICDAR.2013.232 . IEEE

[20] Chu, W.-T., Li, W.-W.: Manga facenet: Face detection in manga based on deep neural network. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 412–415 (2017)

[21] Yanagisawa, H., Yamashita, T., Watanabe, H.: A study on object detection method from manga images using CNN. In: 2018 International Workshop on Advanced Image Technology (IWAIT), pp. 1–4 (2018). https://doi.org/10.1109/IWAIT.2018.8369633 . IEEE

[22] Hartel, R., Dunst, A.: An OCR pipeline and semantic text analysis for comics. In: Del Bimbo, A., Cucchiara, R., Sclaroff, S., Farinella, G.M., Mei, T., Bertini, M., Escalante, H.J., Vezzani, R. (eds.) Pattern Recognition. ICPR International Workshops and Challenges, pp. 213–222. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68780-9_19

[23] Rishu, Kukreja, V.: Decoding comics: a systematic literature review on recognition, segmentation, and classification techniques with emphasis on computer vision and non-computer vision. Multimedia Tools and Applications **84**(24), 28739–28806 (2025) https://doi.org/10.1007/s11042-024-20214-x

[24] Nascimento, T.H., Silva, L., Horbylon, C., Siqueira, D., Felix, J., Aranha, R., Soares, F.: Machine translation of comics with visual reconstruction for linguistic accessibility. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 2076–2081 (2025). https://doi.org/10.1109/COMPSAC65507.2025.00290

[25] Marouvo, G., Pereira, F.: Capturing the narrative: A deep learning pipeline for comics

sequences. In: Gonçalves, N., Oliveira, H.P., Sánchez, J.A. (eds.) Pattern Recognition and Image Analysis, pp. 3–14. Springer, Cham (2026)

[26] Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer, New York (2002). https://doi.org/10.1007/b98835

[27] Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**(2065), 20150202 (2016) https://doi.org/10.1098/rsta.2015.0202

[28] Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, New York (2006). https://doi.org/10.1007/978-0-387-40065-5

[29] Luenberger, D.G., Ye, Y.: Linear and Nonlinear Programming, 3rd edn. Springer, New York (2008). https://doi.org/10.1007/978-0-387-74503-9

[30] Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley, New York (1988). https://doi.org/10.1002/9781118627372

[31] Wolsey, L.A.: Integer Programming. Wiley, New York (1998)

[32] Vielma, J.P.: Mixed integer linear programming formulation techniques. SIAM Review **57**(1), 3–57 (2015) https://doi.org/10.1137/130915303

[33] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

[34] Chu, W.-T., Yu, C.-H., Wang, H.-H.: Optimized comics-based storytelling for temporal image sequences. IEEE Transactions on Multimedia **17**(2), 201–215 (2015) https://doi.org/10.1109/TMM.2014.2383616

[35] Tsubaki, I.: Study of cartoon balloon suitable for comic translation. Reports of the Technical Conference of the Institute of Image Electronics Engineers of Japan **22.04**, 301–302 (2023) https://doi.org/10.11371/wiieej.22.04.0_301

[36] Chu, W.-T., Yu, C.-H.: Optimized speech balloon placement for automatic comics generation. In: Proceedings of the 3rd ACM International Workshop on Interactive Multimedia on Mobile & Portable Devices. IMMPD '13, pp. 1–6. Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2505483.2505486

[37] Dunst, A., Hartel, R., Laubrock, J.: The Graphic Narrative Corpus (GNC): Design, annotation, and analysis for the digital humanities. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 3, pp. 15–20 (2017). https://doi.org/10.1109/ICDAR.2017.286

[38] Goggin, J., Hassler-Forest, D. (eds.): The Rise and Reason of Comics and Graphic Literature: Critical Essays on the Form. McFarland, Jefferson, NC (2014)

[39] Knuth, D.E., Plass, M.F.: Breaking paragraphs into lines. Software: Practice and Experience **11**(11), 1119–1184 (1981) https://doi.org/10.1002/spe.4380111102

[40] Liu, X., Li, C., Wong, T.-T.: Boundary-aware texture region segmentation from manga. Computational Visual Media **3**, 61–71 (2017)

[41] Suzuki, S., Abe, K.: Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing **30**(1), 32–46 (1985) https:

//doi.org/10.1016/0734-189X(85)90016-7

[42] Ortiz, A., Mansfield, J.S., Legge, G.E.: Why does reading speed decrease when text has randomly rotated letters. Investigative Ophthalmology and Visual Science **38**(4), 648 (1997)

[43] Blythe, H.I., Juhasz, B.J., Tbaily, L.W., Rayner, K., Liversedge, S.P.: Reading sentences of words with rotated letters: An eye movement study. Quarterly Journal of Experimental Psychology **72**(7), 1790–1804 (2019)

[44] Todd, M.J.: Minimum-Volume Ellipsoids: Theory and Algorithms. SIAM and Mathematical Optimization Society, Philadelphia (2016). https://doi.org/10.1137/1.9781611974386

[45] Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. SIAM, Philadelphia (2000). https://doi.org/10.1137/1.9780898719857

[46] Adobe Inc.: Adobe Photoshop. https://www.adobe.com/products/photoshop.html

[47] Bradski, G.: The OpenCV library. Dr. Dobb's Journal of Software Tools **25**(11), 120–125 (2000)

[48] Harris, C.R., Millman, K.J., van der Walt, S.J., *et al.*: Array programming with NumPy. Nature **585**, 357–362 (2020) https://doi.org/10.1038/s41586-020-2649-2

[49] Kraft, D.: A software package for sequential quadratic programming. Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, DFVLR-FB 88-28 (1988)

[50] Virtanen, P., Gommers, R., Oliphant, T.E., *et al.*: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods **17**, 261–272 (2020) https://doi.org/10.1038/s41592-019-0686-2

[51] Worcester, K.: Comics, comics studies, and political science. International Political Science Review **38**(5), 690–700 (2017)

[52] OpenAI: ChatGPT, versions 5.1 and 5.2 [large language model]. Exploratory outputs generated in November–December 2025; accessed through ChatGPT (2025)

[53] Cohn, N.: Navigating comics: an empirical and theoretical approach to strategies of reading comic page layouts. Frontiers in Psychology **4**, 186 (2013) https://doi.org/10.3389/fpsyg.2013.00186

[54] Cohn, N., Campbell, H.: Navigating comics II: Constraints on the reading order of comic page layouts. Applied Cognitive Psychology **29**(2), 193–199 (2015) https://doi.org/10.1002/acp.3086