CURSOR: Scalable Mixed-Order Hypergraph Matching with CUR Decomposition

Supplementary Material

6. PRL-based Method Derivation

Consider source graph $\mathcal{G}_1 = \{\mathcal{V}_1, \mathcal{E}_1\}$ and target graph $\mathcal{G}_2 = \{\mathcal{V}_2, \mathcal{E}_2\}$. Since the compatibility matrices/tensors are non-negative in graph matching, the original compatibility coefficient between $(i,l) \in \mathcal{E}_1$ and $(j,m) \in \mathcal{E}_2$, denoted as $r_{il}(j,m) \in [-1,1]$ in [13], is updated to $R_{il}(j,m) = 0.5(r_{il}(j,m)+1)$. The original PRL-based updating becomes

$$p_i^{(k+1)}(j) = \frac{p_i^{(k)}(j)\sum_{l=1}^{n_1}\sum_{m=1}^{n_2} R_{il}(j,m)p_l^{(k)}(m)}{\sum_m p_i^{(k)}(m)\sum_{l=1}^{n_1}\sum_{m=1}^{n_2} R_{il}(j,m)p_l^{(k)}(m)} \tag{15}$$

where $p_i(j)$ represents the probability that i^{th} node of $\mathcal{V}_1$ matches j^{th} node of $\mathcal{V}_2$. As discussed in the main paper, $p_i^{(k)}(j)$ in the numerator plays the role of weight factor. Specifically, if we ignore the effect of $p_i^{(k)}(j)$, Eq. (15) becomes the power-iteration-like linear updating scheme commonly used in SM [21] and TM [9].

Assume the highest probability for $i_0 \in \mathcal{V}_1$ is $p_{i_0}(j_0)$. Hummel and Zucker proved that $p_{i_0}^{(k)}(j_0)$ consistently satisfies $\sum_{l=1}^{n_1}\sum_{m=1}^{n_2} R_{i_0 l}(j_0,m)p_l^{(k)}(m) \geq p_{i_0}^{(k)}(j_0)$ during the updating if $\mathbf{R}$, the matrix form of $R_{i_1 i_2}(j_1,j_2)$ for all (i_1,i_2) and (j_1,j_2), is symmetric [13]. To further accelerate the convergence, the weighting factor is replaced from $p_i^{(k)}(j)$ to $\sum_{l=1}^{n_1}\sum_{m=1}^{n_2} R_{il}(j,m)p_l^{(k)}(m)$, which leads to

$$p_i^{(k+1)}(j) = \frac{[\sum_l \sum_m R_{il}(j,m)p_l^{(k)}(m)]^2}{\sum_j [\sum_l \sum_m R_{il}(j,m)p_l^{(k)}(m)]^2} \tag{16}$$

Define the compatibility coefficients as:

$$R_{i_1 i_2}(j_1,j_2) = \begin{cases} M_{i_1,j_1} & i_1 = i_2 \text{ and } j_1 = j_2 \\ \hat{H}_{i_1,j_1,i_2,j_2} & \text{otherwise} \end{cases} \tag{17}$$

where $M_{i,j}$ is the first-order compatibility between i^{th} node of $\mathcal{V}_1$ and j^{th} node of $\mathcal{V}_2$. $\hat{H}_{i_1,j_1,i_2,j_2}$ denotes the compatibility between $(i_1,i_2) \in \mathcal{E}_1$ and $(j_1,j_2) \in \mathcal{E}_2$. The numerator of Eq. (16) becomes:

$$\hat{p}_i^{(k+1)}(j) = (M_{i,j}p_i^{(k)}(j) + \sum_l \sum_m \hat{H}_{i,l,j,m}p_l^{(k)}(m))^2 \tag{18}$$

$\hat{p}_i^{(k+1)}(j)$ is updated with the combination of first and second-order compatibilities. The main paper shows that for a consistent updating scheme, $\sum_l \sum_m \hat{H}_{i,l,j,m}p_l^{(k)}(m))^2 \geq 0.25M_{i_0,j_0}^{-1}$. By replacing the probability set $\mathbf{p} = \{p_i(j)\}$ with the vector $\mathbf{x}$, the column-wise flattening of soft-constraint assignment matrix $\mathbf{X}$, Eq. (18) can be updated as

$$\hat{\mathbf{x}}^{(k+1)} = (\hat{\mathbf{m}} \odot \mathbf{x}^{(k)} + \mathbf{H}\mathbf{x}^{(k)})^2 \tag{19}$$

where $\odot$ is the element-wise multiplication and the first-order compatibility vector $\hat{\mathbf{m}}$ is obtained by column-wise flattening $\hat{\mathbf{M}}$. The square calculation in Eq. (19) is also element-wise. $\mathbf{H}$ is the second-order compatibility matrix. Since both $\mathbf{H}$ and $\hat{\mathbf{m}}$ are dense, with the all-ones vector as $\mathbf{x}^{(0)}$, Eq. (19) converges consistently.

In our work, the updating scheme for PRL-based hypergraph matching is extended to

$$\hat{\mathbf{x}}^{(k+1)} = (\alpha\hat{\mathbf{m}} \odot \mathbf{x}^{(k)} + (1-\alpha)\mathcal{H} \otimes_1 \mathbf{x}^{(k)} \otimes_2 \mathbf{x}^{(k)})^2 \tag{20}$$

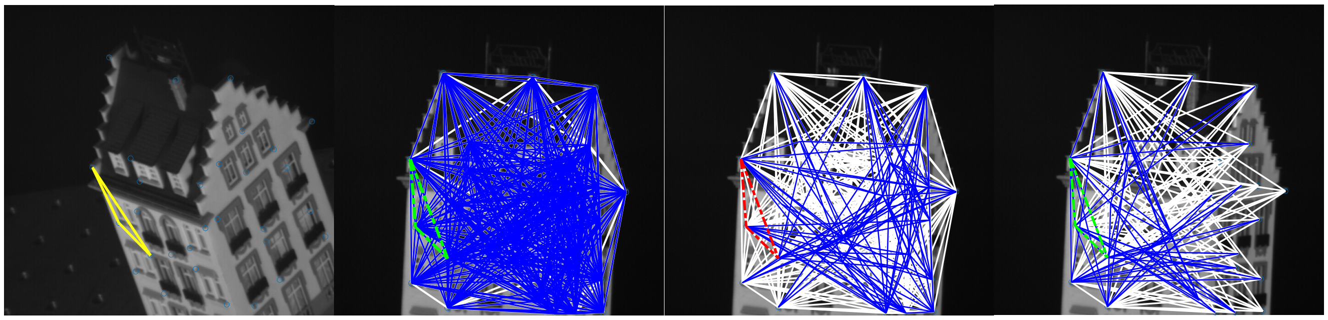
where $\otimes_l$ is the mode-l product of the tensor and vector and $\alpha \in [0,1]$ is a balance weight between the first and third-order compatibilities. Since the third-order compatibility tensor $\mathcal{H}$ is highly sparse, the high value of $\hat{\mathbf{x}}^{(k+1)}$ in Eq. (20) is concentrated if the sparse tensor is reliable. In our work, a reliable tensor means most ground truth hyperedge pair compatibilities are successfully selected in the tensor blocks.

7. Detailed Analysis in Sec. 4

Due to the space limitation of the main paper, we provide a more detailed experiment analysis based on the results of Sec. 4 to discuss the superiority and bottleneck of CURSOR.

7.1. Memory Footprint Analysis in Sec. 4.1

Table 2 shows the detailed memory footprint of the experiment result with CURSOR in Sec. 4.1 of the main paper. Theoretically, the CUR decomposition of the matrix $\mathbf{H}$, requires $O(cn_1n_2)$ space complexity. The tensor $\mathcal{H}$, on the other hand, only needs $O(tr)$. For small-scale problems, the sparse tensor occupies most memory footprint with a small-size matrix. As the graph scale grows, with more columns selected from the compatibility matrices for higher matching accuracy, the main space occupation comes from $\mathbf{H}$, and the second-order CUR-based matching becomes the bottleneck for the graph matching problem. Although CURSOR can deal with larger-scale tasks compared to ANN, its capability to solve scalable problems is limited to the second-order matrix.

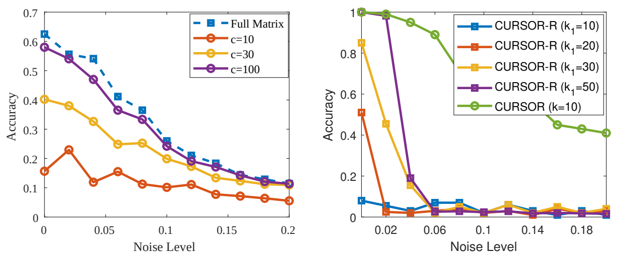


(a) Source (b) ANN ($r_1 = 900$) (c) ANN ($r_1 = 50$) (d) CURSOR ($r = 25$)

Figure 6. The detailed hyperedge correspondences between one sampled hyperedge from the source (yellow triangle in (a)) and target images with ANN ((b) and (c)) and CURSOR (d). The white triangles in the target images denote all the hyperedges compared with the source hyperedge, and the blue triangles are the hyperedges with the highest compatibilities (r_1 for ANN and r for CURSOR). The green dashed triangles represent the matched hyperedges, and the red one represents the mismatch.

Table 2. Detailed memory footprint of CURSOR in Tab. 1 of the main paper.

Problem	Parameter			Memory Footprint
n_1 vs n_2	t	c	r	$\mathbf{H}/(\mathbf{H}+\mathcal{H})$
30 vs 30	900	15	5	0.11MB/0.62MB
30 vs 50	1500	15	5	0.20MB/0.82MB
50 vs 50	2500	20	7	0.40MB/2.37MB
50 vs 100	5000	100	10	3.97MB/9.62MB
100 vs 100	10000	100	20	8.02MB/30.65MB
300 vs 300	30000	200	20	0.14GB/0.21GB
500 vs 500	50000	300	30	0.59GB/0.76GB
800 vs 800	80000	400	50	1.95GB/2.02GB
1000 vs 1000	100000	500	80	4.88GB/5.03GB



(a) (b)

Figure 7. (a) CUR-based second-order graph matching accuracy with various c. (b) The average accuracy using CURSOR with CUR-based pairwise matching result and CURSOR with randomly sampled indices (CURSOR-R).

7.2. Visualization analysis in Sec. 4.3

We provide a more detailed analysis of the experiment results from Sec. 4.3 to demonstrate the superiority of CURSOR, as is shown in Fig. 6. Traditional ANN-based methods compute the compatibilities between the sampled source hyperedges (yellow triangle in Fig. 6a) and all the target ones (fully connected white triangles in Figs. 6b-6c). With the intermediate second-order result, CURSOR computes fewer compatibilities, as is shown in Fig. 6d. To find the corresponding hyperedge (green dashed triangles in Fig. 6), a large amount (the hyperparameter $r_1 = 900$ in Fig. 6b) of the highest compatibilities (blue triangles in Figs. 6b-6d) should be selected for ANN. If we decrease r_1 to 50, some correct ones will be missed (red triangle in Fig. 6c). CURSOR can effectively find the hyperedges with the 25 highest compatibilities (green triangle in Fig. 6d). Compared to the traditional tensor generation methods, CURSOR can effectively increase the matching performance with less computational complexity.

8. Ablation Studies

To further analyze the effectiveness of several design choices in CURSOR, ablation studies were conducted on the 100-vs-110 random synthetic dataset introduced in Sec. 4.1 of the main paper.

8.1. CUR-based Pairwise Matching

We first studied the effectiveness of the CUR-based second-order graph matching. Two experiments were designed to analyze the performance of the pairwise matching with the CUR decomposition of the second-order compatibility matrix $\mathbf{H}$ and the rough intermediate matching result, respectively. The dataset's noise level σ varied in the range $[0, 0.2]$.

The CUR-based second-order graph matching was first evaluated on the dataset with various c, i.e., the number of randomly selected columns from $\mathbf{H}$. During the experiment, only the second-order graph-matching result was analyzed. We compared the proposed method with a PRL-based pairwise matching using the full compatibility ma-

Table 3. Average time consumption for second-order graph matching (in seconds)

Method	Time (s)	
	Computing $\mathbf{H}$	Matching
Full Matrix	6.311	0.491
$c = 10$	0.018	0.016
$c = 30$	0.031	0.016
$c = 100$	0.109	0.043

trix. The matching accuracy is shown in Fig. 7a. Due to the low-rank estimation, the CUR-based method decreased the matching performance. Specifically, as the columns were randomly chosen, the result was quite poor when c was relatively small. However, with less than 1% (100 out of 100×110) of columns selected, the CUR-based pairwise matching algorithm achieved a comparable result to the algorithm using the whole matrix. Table 3 reports the average time consumption of the compatibility matrix generation and graph matching. With only a few columns calculated, the CUR-based second-order algorithm effectively accelerated the matching process.

The effectiveness of the intermediate second-order matching result was further studied. As discussed in the main paper, the second-order matching result $\mathcal{P}^k = \{\mathcal{P}_1^k, \cdots, \mathcal{P}_{n_1}^k\}$ consists k best-matching target nodes for each source node, where n_1 denotes the number of the source nodes. The hypergraph matching result with $\mathcal{P}^k$ was compared to the result with k_1 randomly sampled indices in all three tensor modes (denoted as CURSOR-R). The parameter c and k from $\mathcal{P}^k$ for CURSOR with the pairwise matching result was set as 100 and 10 respectively. For CURSOR-R, k_1 varied from 10 to 50. For both methods, the number of randomly selected hyperedges from the source hyperedges $t = 3000$, the number of highest compatibilities in each tensor block $r = 100$, and the balance factor of PRL-based algorithm α was set to 0.2. As shown in Fig. 7b, CURSOR-R had a lower matching accuracy even for $k_1 = 50$. Assuming the number of nodes is n_2 in the target graph, theoretically, only 3 out of $3n_2^2$ fibers for each tensor block, i.e., key fibers in our work, contain the entry of the ground truth hyperedge pair. Due to the random sampling, the probability of selecting the key fiber is $(k_1/n_2)^2$, around 20.7% when $k_1 = 50$ and $n_2 = 110$. Therefore, the sparse compatibility tensor generated by CURSOR-R was highly unreliable. CURSOR with the CUR-based pairwise matching result selected the key fiber in each tensor block with a high probability, effectively increasing the final accuracy.

8.2. CURSOR vs ANN-based Tensor Generation

Experiment results in Sec. 4 of the main paper have already shown that the proposed PRL-based algorithm with CURSOR achieved higher matching accuracy than other algorithms in most cases. One may wonder how the ANN-based tensor generation performs with the same hypergraph matching algorithm. In this experiment, we further compare CURSOR with the ANN-based tensor generation method applying PRL-based algorithm on the 100-vs-110 random synthetic dataset. During experiment, $c = 100$, $k = 10$ and $r = 100$ for CURSOR. For ANN-based tensor generation, r_1 was set as 100 for the same tensor density. For both methods, $t = 3000$ and $\alpha = 0.2$. The stopping criterion of the PRL-based algorithm was set as $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2 \leq 10^{-8}$ and the maximum number of iteration was 100.

The average matching accuracy is reported in Fig. 8a. The ANN method shows an unstable performance with high σ since the compatibility tensor was too sparse. When α was low, since the PRL-based algorithm focused on the third-order compatibilities, the ANN method did not generate a reliable compatibility tensor. To make the algorithm focus more on first-order compatibilities, α was further increased to 0.8 for the ANN case, which significantly improved the accuracy. The average number of iterations to converge is shown in Fig. 8b. When $\sigma > 0.02$, the ANN method did not converge within 100 iterations. With the same tensor sparsity, CURSOR successfully converged in all the cases. The decay of $\mathbf{x}^{(k)}$ per iteration for the ANN method and CURSOR with $\sigma = 0.1$ was further analyzed, as shown in Fig. 8c. The ANN method did not converge due to few ground truth hyperedge compatibilities selected from the whole tensor. However, CURSOR chose the nonzero compatibilities from a smaller reliable searching region. Therefore, it is capable of converging fast with the same tensor sparsity.

9. Parameter Sensitivity Analysis

Experiments below are provided to investigate how the hyperparameters in CURSOR affect the final results. Since the hyperparameter t, the number of randomly selected hyperedges, was thoroughly studied in previous works [9, 19, 20], we do not redundantly analyze it here. The method was evaluated on the random 100-vs-110 synthetic dataset introduced in Sec. 4.1 of the main paper as well.

9.1. Parameter in Second-order Graph Matching

The sensitivity of parameter c to the whole framework was first analyzed. During the experiment, the whole compatibility matrix was calculated directly with noise level $\sigma = 0.02$. CUR decomposition was evaluated on the matrix with various c for the pairwise graph matching. To analyze the influence of c on the second-order matching result, we define hit rate, calculated as $\sum_{i=1}^{n_1} \delta_i / n_1$, where

$$\delta_i = \begin{cases} 1 & \text{The true match } j \in \mathcal{P}_i^k \\ 0 & \text{Otherwise} \end{cases} \tag{21}$$

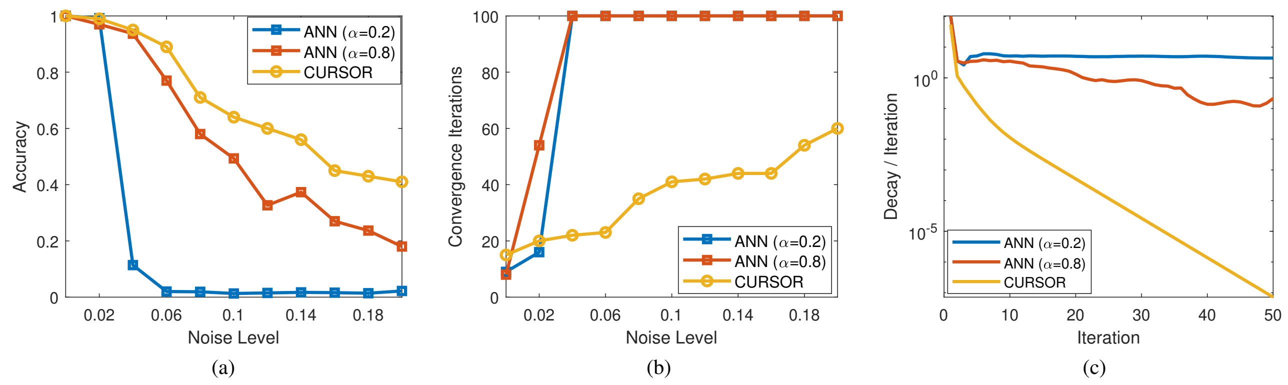
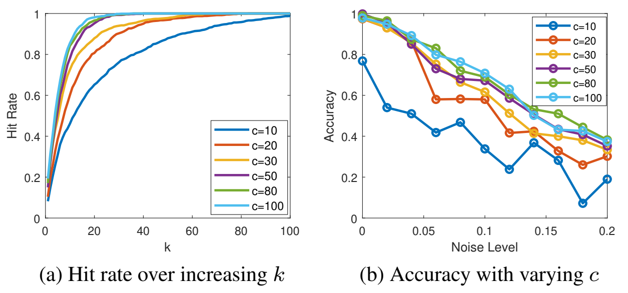


Figure 8. Results on 100-vs-110 synthetic dataset comparing CURSOR with ANN, $\alpha = 0.2$ for CURSOR. (a) The average matching accuracy using CURSOR and ANN with different α. (b) The average iterations for convergence using CURSOR and ANN with different α. (c) The decay $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2$ per iteration with different α using CURSOR and ANN. The noise level $\sigma = 0.1$.



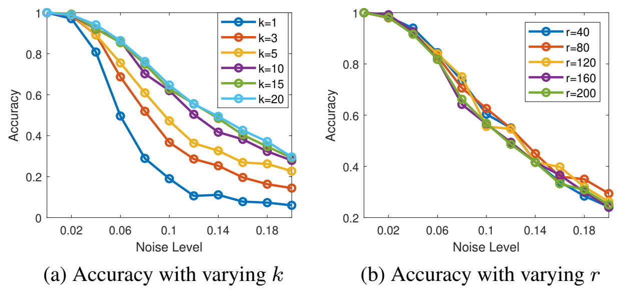
(a) Hit rate over increasing k (b) Accuracy with varying c

Figure 9. (a) The average hit rate over increasing k on a 100-vs-110 synthetic dataset with $\sigma = 0.02$ from the CUR-based second-order graph matching results. (b) The average accuracy on the 100-vs-110 synthetic dataset with $\sigma \in [0, 0.2]$ and $c \in [10, 100]$.

The average hit rate was computed over the increasing number of selected highest compatibilities k, as shown in Fig. 9a. Each curve represents the hit rate with c during CUR decomposition. To achieve the same hit rate and set k as small as possible, theoretically, the number of selected columns needed to be as large as possible. However, when increasing c from 50 to 100, the gain on hit rate is minor, with a relatively small k to reach a promising hit rate like 0.9. Therefore, to balance the time consumption and the matching performance, a relatively small proportion of columns is sufficient for the rough pairwise matching result.

The influence of c on the final matching result was further analyzed. During the experiment, we set $k = 10$, $r = 100$, and $\alpha = 0.2$. The noise level of the dataset was assigned as $\sigma \in [0, 0.2]$. The result is reported in Fig. 9b. Since the columns of the compatibility matrices were randomly selected, the matching result was unstable when c was less than 20. The performance gradually saturated with over 50 columns (around 0.5% of the total columns). The experiment results in Fig. 9a show that 50-100 columns can already provide a reliable second-order matching result for the following higher-order process, effectively decreasing the computation cost.

9.2. Parameters During Tensor Generation

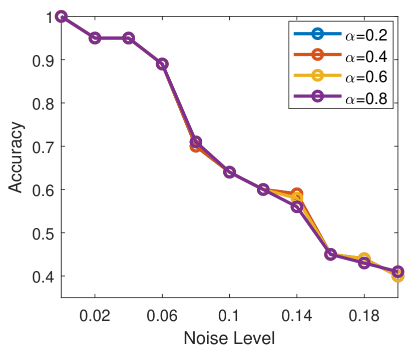


(a) Accuracy with varying k (b) Accuracy with varying r

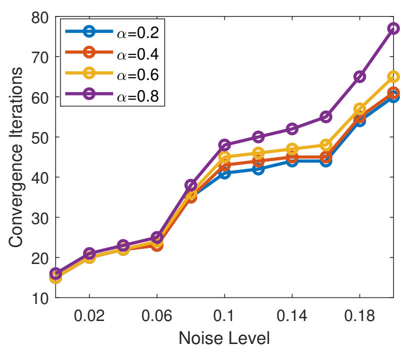
Figure 10. Results on 100-vs-110 synthetic dataset with $\sigma \in [0, 0.2]$ using CURSOR. (a) The matching accuracy with increasing k. (b) The matching accuracy with increasing r.

The sensitivity of k and r to the final matching accuracy was further studied. During the experiment, c was set as 100 and $\alpha = 0.2$.

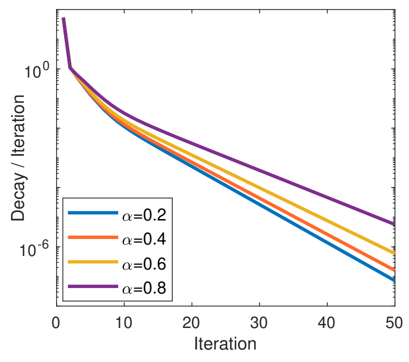
To figure out the influence of k, the parameter r was set as 100, and k varied from 1 to 20. The result is shown in Fig. 10a. It is obvious that a higher k can achieve higher robustness for target points with high noise impact. However, with a polynomial $O(tk^2n_2)$ computation cost for tensor generation, the matching performance gradually reached its peak. For instance, the matching accuracy increased more than 35% from $k = 1$ to $k = 5$ when $\sigma = 0.08$, but less than 2% from $k = 15$ to $k = 20$. For each tensor block, if the entry of the ground truth paired hyperedge compatibility was included in r non-zero elements, the corresponding nodes would be matched with a high probability. Therefore, an appropriate k needs to be set to balance the computation



(a)



(b)



(c)

Figure 11. Results on 100-vs-110 synthetic dataset with PRL-based matching algorithm. (a) The average matching accuracy using CURSOR with different α. (b) The average iterations for convergence using CURSOR with different α. (c) The decay $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2$ per iteration with different α settings using CURSOR. The noise level $\sigma = 0.1$.

cost and the performance in practical use.

The effect of r was further analyzed by setting k as 10 and varying r from 20 to 200, as shown in Fig. 10b. Unlike the results in Fig. 10a, CURSOR with the highest r performed the worst. The reason may be that when the ground truth hyperedge pair is already included in the non-zero compatibilities, more redundant compatibilities can cause lower matching performance.

9.3. Parameters of PRL-based Matching Algorithm

To study the sensitivity of α, the matching accuracy and convergence speed using CURSOR were further analyzed. During the experiment, $c = 100$, $k = 10$, and r was set as 100. The stopping criteria of the PRL-based algorithm was assigned as $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2 \leq 10^{-8}$. The matching accuracy with various α is shown in Fig. 11a. With a reliable compatibility tensor, the PRL-based matching algorithm achieved almost the same performance regardless of α. We further analyzed their convergence speeds, as shown in Figs. 11b and 11c. As the noise level increased, more iterations were required to satisfy the stopping criteria, and a lower α achieved faster convergence. As discussed in the previous sections, the parameter α is a balanced factor between first and third-order compatibilities. Although the method converged faster with $\alpha = 0$, the first-order compatibilities stabilized the matching process and increased the matching performance under some extreme circumstances.